Replication of Data Placement for Uncertain Scheduling

Manmohan Chaubey, Erik Saule
Department of Computer Science
University of North Carolina at Charlotte
Charlotte, USA
Email: mchaubey@uncc.edu, esaule@uncc.edu

***Abstract*—The abstract goes here. DO NOT USE SPECIAL CHARACTERS, SYMBOLS, OR MATH IN YOUR TITLE OR ABSTRACT.**

***Index Terms*—component; formatting; style; styling;**

I. Introduction

Many real world scheduling problems related to task allocation in parallel machines are uncertain in nature. The fact makes the class of problem known as 'scheduling with uncertainty' or 'robust' scheduling an important and most studied problem in Scheduling. Often in scheduling models the exact value of parameters such as processing times of tasks, are not known initially, but they have a range outside which their values cannot lie. The paper motivates the task uncertainty and data placement in scheduling in parallel systems. The scheduling problem related to data placement and task allocation is common in heterogeneous systems and often dealt with common approach. The objective of this work is to propose an optimization approach that takes into account an effective data placement and scheduling of tasks by replication in the environment where processing time of tasks are not known exactly but they can be estimated based on some pre estimated values. The paper is important in the sense that it proposes different models depending upon different scenarios and compares them based on approximation ratio and replication they allow. Replication strategy that allows to place data in a wisely manner offers a faster access to files required by jobs, hence increases the job execution's performance. Replication helps in load balancing but it have certain cost attached with it. The paper also deals with this problem and chooses the scenario in which replication is beneficial.

II. Problem Definition

We have set J of n jobs which need to be scheduled to set M of m machines such that makespan, C_{max} is minimized. C^*_{max} denotes optimal makespan of schedule. We are considering the problems where the scheduler do not know the processing time of a task exactly before it completes, but we have some estimation of the processing time before the task is assigned to a processor. We know that the actual processing time of a task i is between $\frac{1}{\alpha}$and α times of its estimated processing time. p_i denotes the actual processing time and $\tilde{p_i}$ denotes estimated processing time for the task i. We have this estimate:

$$\frac{1}{\alpha} \leq \frac{p_i}{\tilde{p_i}} \leq \alpha \qquad (1)$$

We consider various problem models incorporating the above mentioned scenario of uncertain processing times of tasks with an estimate. Each problem is constructedin two phases. In first phase we choose where data is replicated using only $\tilde{p_i}$. In second phase we choose actual shedule with semi-clairvoyant algorithm which uses only approximate knowledge of initial processing time and after scheduling the task the actual p_i is known. With objective to find schedule which minimizes makespan, we investigate greedy algorithms for each of the problem models and prove their comptetitive ratios.

We have used Graham's List Scheduling (LS) and Largest Processing Time (LPT) algorithms to derive approximation ratios in different scenarios. The LS algorithm takes tasks one at a time and assign it to the processor having least load at that time. LS is 2-approximation algorithm and is widely used in online scheduling problems. The LPT sorts tasks in decreasing order of processing time and assign them one at a time in this order to the processor with the smallest current load. The LPT algorithm have worst case performance ratio as

$4/3 - 1/(3m)$ in offline setting . Depending upon which among these two algorithms suits more for a problem model we have used these algorithms accordingly.

III. Model 1: No replication

In this problem model we have considered situation where each task is restricted to be scheduled on only one machine. We have a set J of n jobs, and a set M of m machines. The processing time of job j is p_j on machine i. Let $f : J \leftarrow M$ be a function that assigns each job to exactly one machine. Let us denote E_i as the set tasks which is assigned to a machine i. The restriction that each task can be scheduled to only one machine restrict the problem constrution to phase one only. There is no replication of tasks in this model.

We have considered List scheduling and Longest processing time algorithms and have assigned each task on machine on which they are restricted to. Based on the estimated processing times of tasks loaded on each machine we will derive makespan.

A. Lower Bound

Lemma 1.1: There is no online algorithm having competitive ratio better than α^2.
Proof: We use adversery technique to prove that there is no online algorithm having competitive ratio better than α^2

Let total tasks be λM with $\tilde{p_i} = 1$. After scheduling let the most loaded machine j have B tasks. So, $B \geq \lambda$. Adversery technique increases the task on j by α and decreases the other task by α. So, C_{max} becomes αB and $C^*_{max} \geq \frac{\alpha B + \frac{1}{\alpha}(\lambda M - B)}{M}$. We have,

$$\frac{C_{max}}{C^*_{max}} \leq \frac{\alpha^2 BM}{\alpha^2 B + \lambda M - B} = \frac{\alpha^2 M}{\alpha^2 + \frac{\lambda M}{B} - 1}$$

So, the value of above expression is directly proportional to B. For $B = \lambda$ the value of expression would be minimum.

B. Using LPT

In this problem model we are assigning tasks to processors using LPT in offilne mode using non-inceasing order of estimated processing times of tasks.

$$\tilde{C}_{max} \leq [\frac{\sum \tilde{p}_j + (m-1)\tilde{p}_l}{m} \quad (2)$$

where $\tilde{C}_{max}$ is makespan considering esimated processing times of tasks. Also actual processing time C_{max} must be smaller the $\alpha\tilde{C}_{max}$, we have

$$C_{max} \leq \alpha\tilde{C}_{max} \leq \alpha[\frac{\sum \tilde{p}_j + (m-1)\tilde{p}_l}{m}] \quad (3)$$

Considering worst case situation the $\tilde{C}_{max}$ will increase to α times and load on rest of the processors will shrink $\frac{1}{\alpha}$ times. So total actual prosessing time will be given by following equation. The argument behind this is that greater the value of ratio $\frac{C_{max}}{\sum p_j}$, the algorithm will give bad approximation. So increasing the load on machine wchich have $C_{max}$to maximum value and decrasing rests of the load on other processors will reach worst case scenario. So total actual processing time will be given by following equation.

$$\sum p_j = \frac{\tilde{p}_j - \tilde{C}_{max}}{\alpha} + \alpha\tilde{C}_{max} \quad (4)$$

Also the actual optimal processing time have following constraint

$$C^*_{max} \geq \frac{\sum p_j}{m}$$

After putting the value of $\sum p_j$, we have

$$mC^*_{max} \geq \frac{\tilde{p}_j - \tilde{C}_{max}}{\alpha} + \alpha\tilde{C}_{max}$$

$$\Rightarrow mC^*_{max} \geq \frac{\tilde{p}_j - [\frac{\sum \tilde{p}_j + (m-1)\tilde{p}_l}{m}]}{\alpha} + C_{max}$$

$$mC^*_{max} \geq \frac{m-1}{\alpha m}[\sum \tilde{p}_j - \tilde{p}_l] + C_{max}$$

By the property of LPT $\sum \tilde{p}_j - \tilde{p}_l \geq m(\tilde{C}_{max} - \tilde{p}_l)$, putting this we have,

$$mC^*_{max} \geq \frac{m-1}{\alpha}[\tilde{C}_{max} - \tilde{p}_l] + C_{max}$$

$$\Rightarrow mC^*_{max} \geq \frac{m-1}{\alpha}[\tilde{C}_{max} - \frac{\tilde{C}_{max}}{2}] + C_{max}$$

$$\Rightarrow mC^*_{max} \geq \frac{m-1}{2\alpha}[\frac{C_{max}}{\alpha}] + C_{max}$$

$$\Rightarrow mC^*_{max} \geq [\frac{m-1}{2\alpha^2} + 1]C_{max}$$

$$\Rightarrow \frac{C_{max}}{C^*_{max}} \leq \frac{2\alpha^2 m}{2\alpha^2 + m - 1}$$

IV. Model 2: replication is done everywhere

In this problem model we consider no restrictaion on task assignment. There is no initial first phase to select where replication need to be done. All tasks are allowed to replicate everywhere There are n tasks which need to be assigned to m processors. We have applied simple Largest Processing Time Algorithms (LPT) using estimated processing times of tasks in non-increasing order as input.

Theorem 2.1: $C^*_{max} \geq \frac{2}{\alpha^2} p_l$ for when there are at least two tasks in the machine to which l is scheduled.

Proof: Suppose l be the last task with estimated processing time $\tilde{p}_l$. Suppose there are at least two tasks in the machine in which l is assigned including l. Let C_{max} be the be makespan of the schedule and C^*_{max} be the optimal makespan.

As there is at least one task j before l in the machine to which l is assigned, we have

$$C^*_{max} \geq p_l + p_j$$

As actual processing time of a task must be greater than $\frac{1}{\alpha}$ times of its estimated value, we have

$$C^*_{max} \geq \frac{1}{\alpha}\tilde{p}_l + \frac{1}{\alpha}\tilde{p}_j$$

As j is scheduled before l using LPT on estimated values of processing times, $\tilde{p}_j \geq \tilde{p}_l$ holds true for tasks l and j. Using this, we have

$$C^*_{max} \geq \frac{2}{\alpha}\tilde{p}_l$$

$$\Rightarrow C^*_{max} \geq \frac{2}{\alpha^2} p_l$$

So, for at least two tasks in machine, We have the following condition

$$C^*_{max} \geq \frac{2}{\alpha^2} p_l \tag{5}$$

Theorem 2.2: $\frac{C_{max}}{C^*_{max}} \leq 1 + (\frac{m-1}{m})\frac{\alpha^2}{2}$

proof: The optimal makespan, C^*_{max} must be atleast equal to average load on the m machines, We have

$$C^*_{max} \geq \frac{\sum p_j}{m} \tag{6}$$

By the property of LPT the load on each machine i is greater than the load on machine which reach C_{max} before the last task l is scheduled. So for each machine, $C_{max} \leq \sum_{j \in E_i} p_j + p_l$ holds true. Adding for all the machines, We have

$$mC_{max} \leq \sum p_j + (m-1)p_l$$

$$C_{max} \leq \frac{\sum p_j}{m} + \frac{(m-1)}{m} p_l \tag{7}$$

Using (8) and (9), We have

$$\frac{C_{max}}{C^*_{max}} \leq 1 + \frac{m-1}{m}\left(\frac{p_l}{C^*_{max}}\right)$$

As $C^*_{max} \geq \frac{2}{\alpha^2} p_l$, We have

$$\frac{C_{max}}{C^*_{max}} \leq 1 + \left(\frac{m-1}{m}\right)\frac{\alpha^2}{2}$$

Hence the theorem follows.

V. Model 3: Replication in groups

We are tackling the problem of restricted scheduling where tasks are scheduled to machines in two phases. The first phase is in offline mode and each task is pre-assigned to a particular group of processors. In the second phase the tasks are scheduled within the group they are assigned to in first phase. We have a set J of n jobs. The size of each group is equal and have m processors within each group. We have considered task allocation in a group such that each task can be assigned to only one group. In phase 2 each task is scheduled to a particular processor within the group it was allocated in phase 1. We propose List Sheduling algorithm in both the phases. In phase 1 using LS we pre-assign the tasks in different groups. In phase 2 we use online LS to schedule tasks to processors within each group.

A. When number of groups is 2

Theorem 3.1: When number of groups is 2 the approximation ratio is $1 + \frac{2\alpha^2}{1+\alpha^2}$

Proof: Let us consider two groups be $G1$ and $G2$ and after phase 1 the load on each of two processors be $\sum_{i \in G1} \tilde{p}_i$ and $\sum_{i \in G2} \tilde{p}_i$ respectively. Without loss of generality, We can assume makespan, C_{max} comes from $G1$. As the consequense of List Scheduling load difference between two groups cannot exceed estimated longest task, $max_{i \in T} \tilde{p}_i$

$$|\sum_{i \in G1} \tilde{p}_i - \sum_{i \in G2} \tilde{p}_i| \leq max_{i \in T} \tilde{p}_i$$

$$\Rightarrow \sum_{i \in G2} \tilde{p}_i \geq \sum_{i \in G1} \tilde{p}_i - max_{i \in T} \tilde{p}_i$$

As the actual processing time of a task is between $\frac{1}{\alpha}$ and α times of its estimated value, so we have the following inequality

$$\Rightarrow \alpha \sum_{i \in G2} p_i \geq \frac{1}{\alpha} \sum_{i \in G1} p_i - \alpha max_{i \in T} p_i$$

$$\Rightarrow \sum_{i \in G2} p_i \geq \frac{1}{\alpha^2} \sum_{i \in G1} p_i - max_{i \in T} p_i \quad (8)$$

In phase 2, We apply List Scheduling in online mode. As makespan, C_{max} comes from $G1$. Using List Scheduling, We have

$$C_{max} \leq \frac{\sum_{i \in G1} p_i}{m} + \frac{m-1}{m} p_{max} \quad (9)$$

where p_{max} is the longest task in $G1$

Also, C^*_{max} must be greater than the average of the loads on machines, We have

$$C^*_{max} \geq \frac{\sum_{i \in T} p_i}{2m}$$

$$\Leftrightarrow C^*_{max} \geq \frac{\sum_{i \in G1} p_i}{2m} + \frac{\sum_{i \in G2} p_i}{2m}$$

from (10), We derive

$$C^*_{max} \geq \frac{\sum_{i \in G1} p_i}{2m} + \frac{1}{2m\alpha^2} \sum_{i \in G1} p_i - \frac{1}{2m} max_{i \in T} p_i$$

$$\Rightarrow 2m\alpha^2 [C^*_{max} + \frac{max_{i \in T} p_i}{2m}] \geq \alpha^2 \sum_{i \in G1} p_i + \sum_{i \in G1} p_i = (1+\alpha^2) \sum_{i \in G1} p_i$$

$$\Rightarrow \sum_{i \in G1} p_i \leq \frac{2m\alpha^2}{1+\alpha^2} [C^*_{max} + \frac{max_{i \in T} p_i}{2m}] \quad (10)$$

Using (11)and (12),
$C_{max} \leq \frac{2\alpha^2}{1+\alpha^2} [C^*_{max} + \frac{max_{i \in T} p_i}{2m}] + \frac{m-1}{m} p_{max}$

As $C^*_{max} \geq max_{i \in T} p_i \geq p_{max}$, we have

$C_{max} \leq \frac{2\alpha^2}{1+\alpha^2} [C^*_{max} + \frac{C^*_{max}}{2m}] + \frac{m-1}{m} C^*_{max}$

So we have approximation ratio,
$\frac{C_{max}}{C^*_{max}} \leq \frac{2\alpha^2}{1+\alpha^2} + \frac{\alpha^2}{m(1+\alpha^2)} + \frac{m-1}{m}$

$\leq 1 + \frac{2\alpha^2}{1+\alpha^2}$

B. When number of groups is k

Theorem 3.2: When the number of groups is k the approximation ratio is $\frac{\alpha^2}{\alpha^2+k-1} [k + \frac{k-1}{m}] + \frac{m-1}{m}$

Proof: We have k groups of m machines. We have restriction that each task can be assigned to only one of these groups.

Let us consider that C_{max} comes from group $G1$. Also, taking the property of List Scheduling that the load difference between any two groups cannot be greater than the largest task. So, for any group $Gl \neq G1$, We have

$|\sum_{i \in G1} \tilde{p}_i - \sum_{i \in Gl} \tilde{p}_i| \leq max_{i \in T} \tilde{p}_i$ for all, $l = 2, 3, ..., k$

Adding for all values of l, We have

$|(k-1) \sum_{i \in G1} \tilde{p}_i - \sum_{l=2}^{k} \sum_{i \in Gl} \tilde{p}_i| \leq (k-1) max_{i \in T} \tilde{p}_i$

$5 \Rightarrow \sum_{l=2}^{k} \sum_{i \in Gl} \tilde{p}_i \geq (k-1)[\sum_{i \in G1} \tilde{p}_i - max_{i \in T} \tilde{p}_i]$

As the actual processing time of tasks can vary within a factor α and $\frac{1}{\alpha}$ of their estimated processing time, the following inequality holds

$\alpha \sum_{l=2}^{k} \sum_{i \in Gl} p_i \geq (k-1)[\frac{1}{\alpha} \sum_{i \in G1} p_i - \alpha max_{i \in T} p_i]$

$$\sum_{l=2}^{k} \sum_{i \in Gl} p_i \geq (k-1)[\frac{1}{\alpha^2} \sum_{i \in G1} p_i - max_{i \in T} p_i] \quad (11)$$

In phase 2, We are applying LS on online mode. We assume that C_{max} comes from $G1$. Using LS property we can write,

$$C_{max} \leq \frac{\sum_{i \in G2} p_i}{m} + \frac{m-1}{m} p_{max} \quad (12)$$

Also, C^*_{max} must be greater than the average of the loads on machines.

$C^*_{max} \geq \frac{\sum_{i \in T} p_i}{km}$

$\Rightarrow C^*_{max} \geq \frac{\sum_{i \in G1} p_i + \sum_{l=2}^{k} \sum_{i \in Gl} p_i}{km}$

from (13), we derive
$\Rightarrow C^*_{max} \geq \frac{\sum_{i \in G1} p_i + (k-1)[\frac{1}{\alpha^2} \sum_{i \in G1} p_i - max_{i \in T} p_i]}{km}$

$\Rightarrow km\alpha^2 C^*_{max} + \alpha^2(k-1) max_{i \in T} p_i \geq \alpha^2 \sum_{i \in G1} p_i + (k-1) \sum_{i \in G1} p_i$

$$\frac{\alpha^2}{\alpha^2 + k - 1}[kmC^*_{max} + (k-1)max_{i \in T} p_i] \geq \sum_{i \in G1} p_i \quad (13)$$

Using (14) and (15), We have
$C_{max} \leq \frac{\alpha^2}{\alpha^2+k-1}[kC^*_{max} + \frac{(k-1)}{m} max_{i \in T} p_i] + \frac{m-1}{m} p_{max}$

As $C^*_{max} \geq max_{i \in T} p_i \geq p_{max}$, we have

$C_{max} \leq \frac{\alpha^2}{\alpha^2+k-1}[kC^*_{max} + \frac{k-1}{m} C^*_{max}] + \frac{m-1}{m} C^*_{max}$

So we have approximation ratio,
$\frac{C_{max}}{C^*_{max}} \leq \frac{\alpha^2}{\alpha^2+k-1}[k + \frac{k-1}{m}] + \frac{m-1}{m}$

VI. Conclusion

The conclusion goes here. this is more of the conclusionhhhhhhh

Acknowledgment

The authors would like to thank... more thanks here

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.