Replicated 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*—Scheduling problems is a common theoretical tool to analyze the performance of distributed computing systems, such as their load balance. How to distribute the input data to be able to execute a set of tasks in the minimum amount of time can be modeled as a scheduling problem. Often these models assume that the computation time required for each task is known accurately. However in many practical case, only approximate values are available at the time of scheduling.**

In this paper, we investigate how replicating the data required by the tasks can help coping with the inaccuracies of the processing times. In particular, we investigate the problem of scheduling independent tasks to optimize the makespan on a parallel system where the processing times of tasks are only known up to a multiplicative factor. The problem is decomposed in two phases: a first offline phase where the data of tasks are placed and a second online phase where the tasks are actually scheduled.

For this problem we investigate three different strategies, each allowing a different degree of replication of jobs: a) No Replication b) Replication everywhere and c) Replication in groups. We propose approximation algorithms and theoretical lower bound on achievable approximation ratios. This allows us to study the trade-off between the number of replication and the guarantee on the makespan.

***Keywords*-Scheduling; Uncertainty; Robustness; Replication; Approximation Algorithms; Parallel System**

I. Introduction

Many real world scheduling problems related to task allocation in parallel machines are uncertain in nature. This fact makes the class of problem known as 'scheduling with uncertainty' or 'robust' scheduling an important and most studied problem in Scheduling. Robustness is the measure of to what extent an algorithm can cope with uncertainties in a scheduling problem. Often in scheduling problems the exact processing time of a job is not known initially at the time of planning for job- allocation to different machines, but they have a range outside which their values cannot lie. The fact gives a basis to estimate the processing times and to take job assignment decision based on these estimated values. The actual processing time of a job can vary drastically from its estimated one. So, any algorithm have to incur the cost of estimation which impacts its performance negatively. A robust approach deals with uncertainty in input parameters to minimize this cost.

The objective of our research is to study the affect on performance of parallel machine scheduling when jobs are scheduled based on their estimated values of processing times, and to propose scheduling approaches that optimizes the performance in this environment. Data placement and replication techniques can be useful in the scenario of uncertain processing times of tasks. Effective data placement using replication strategy allows better load balancing and hence reduces turnaround job time. This 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 cost attached with it as it usually increases resource usage [17]. The paper deals with this problem and chooses the scenario in which replication is beneficial. Thus our research focus on two area of scheduling problem i.e. task uncertainty and replication using data placement. The scheduling problem related to data placement and task allocation is common in heterogeneous systems and often dealt with common approach. This paper draws qualitative analysis of the effect of replication in data placement for uncertain scheduling with inexact processing time of a task.

The remaining of the paper is organized as follows: we describe system model and notations in section II. Related works are touched in section III. Sections IV-VI describes the 3 different strategies- a)No replications, b)replication is done everywhere and c)replication is done within a group. The respective sections describe derivation of competitive ratios for each of these strategies. Section VII summarizes the results of 3 models based on experimental results.

II. Problem Definition

Let J be a set of n jobs which need to be scheduled onto a set M of m machines. We will use interchangeably the terms machines and processors. Also we will use interchangeably the terms jobs and tasks. We are considering the problems where the scheduler does not know the processing time p_i of task i exactly before the task completes. But the scheduler has access to some estimation of the processing time $\tilde{p_i}$ of task i before making any scheduling decisions. We assume that the actual processing time p_i of a task i is within a multiplicative factor α of the estimated processing time $\tilde{p_i}$. α is a quantity known to the scheduler. In other words the scheduler knows that:

$$\frac{1}{\alpha} \leq \frac{p_i}{\tilde{p_i}} \leq \alpha \tag{1}$$

Assuming that the processing time of the tasks is known to be in an interval is reasonable in many application scenarios. One could derive bounds experimentally using machine learning techniques: for instance [21] used support vector machines to predict the time it will take to run graph traversal algorithms. Also models of runtime of algorithms can be derived analytically: in [7] the authors provide bounds for the performance of various sparse linear algebra operations using only the size of the matrices and vector involved.

The problem is to optimize the makespan, C_{max}. C^*_{max} denotes optimal makespan of a schedule S.

We consider various problem models incorporating the above mentioned scenario of uncertain processing times of tasks with an estimate. Depending upon a problem model the scheduling is done in two phases or just in one phase. In Phase 1, we allocate tasks to certain processors or group of processors depending upon a problem model. Phase 1 chooses where data to be replicated using estimated processing time $\tilde{p_i}$, for each of the task i. Phase 1 takes $\tilde{p_i}$, m and α as input and outputs set of machines, $M_j \subseteq M$ where a task j can be scheduled or is replicated. Phase 2 takes output of phase 1 as input and outputs set of task assigned to a machine i, $E_i \subseteq J$. In Phase 2, we choose actual schedule 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 competitive ratios.

We have used Graham's List Scheduling (LS) [**?**] and Largest Processing Time (LPT) algorithms [9] 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. Related Work

We begin with the literature on scheduling problems under uncertainty. Based on various models of describing uncertainty in input parameter, uncertainty problems can be approached by various methodologies including Reactive, stochastic, fuzzy and Robust approach [11]. The bounded uncertainty model assumes that an input parameter have value between a lower and upper bound and is usually dealt with robust approach or sensitivity analysis. We are more interested in robust approach to deal with uncertainty. There is very less literature available for scheduling with estimated processing time of a task. Most of the literature till now focuses on job size estimation with small error. Very less work is done which focuses on coping with less restrictive estimation for task size and which analyzes the effect of estimation on scheduling quantitatively. Wierman and Nuyens [19] introduce SMART as classification to understand size based policies and draw analytical co-relation between response time and estimated job size in single server problem. Dealing with uncertainty problems with robust approach is widely used in practicality in the area of MapReduce [10] [16], Hadoop [20] [18], databases [12] and web servers [3]. HSFS and FLEX schedulers are proposed which provides robustness in scheduling against uncertain Job Size [20] [13]. Cannon and Jeannot [2] provides scheduling heuristics that optimizes both makespan and robustness in scheduling task graph on heterogeneous system.

Most of the work on robust scheduling uses scenarios in problem formulation to structure variability of uncertain parameter. Daniels and Kouvelis [5] used scenarios to formulate branch and bound based robust scheduling to cope uncertainty in processing time of jobs in single machine. Davenport, Gefflot, and Bek used slack based technique to add extra time to cope with uncertainty [6]. Gatto and Widmayer derives bounds on competitive ratio of Graham's online algorithm in scenario where processing times of jobs either increase or decrease arbitrarily due to perturbations [8]. Their notion of

perturbed processing times is similar to our definition of estimated processing times which can increase or decrease to actual processing times once the jobs are processed. But unlike our work they considered increment and decrement of job processing times as different problem scenario. We have approached the problem with worst case scenario where some tasks may increase and some may decrease in the same schedule.

We have used pro-active approach to deal with uncertainty. Through this research we study effect of load balancing in scenario of uncertain processing times. Based on different task assignment criteria we propose 3 models for our problem definition, offering different degree of task replication for load balancing. Data placement and replication methodologies are highly used in distributed systems including peer-to-peer and Grid systems to achieve effective data management and improve performance [4][1][14]. Our notion of using replication is to increase data availability thereby enhancing system reliability against uncertainties. With focus on studying replication affect on performance of a schedule under uncertainty, we have not considered cost of replication in terms of memory utilization. Our work considers an ideal case where storage is sufficient. Similar work is done by Tse [15] who used selective replication of documents in bi criteria problem of minimizing load and memory in distributed file servers.

A. Our Contribution

We study the effect of load balancing through replication in classic problem of scheduling jobs with uncertain processing times with objective to optimize makespan. We propose 3 models based on different degree of of replication they allow: a)No Replication i.e. $|M_j| = 1$,b) Replication is done every where i.e.$|M_j| = |M|$ and c) Replication in groups i.e $|M_j| = m/k$ where k is number of groups each having m/k processors . For each of the models we propose a two phase algorithm based on Graham's LS and LPT; and derive performance ratio for each. For $|M_j| = 1$, we prove that there is no algorithm which give better performance than α^2. Our algorithm for the model give $\frac{2\alpha^2 m}{2\alpha^2+m-1}$ approximation. For models with replications $|M_j| = |M|$ and $|M_j| = m/k$ the algorithms give performance ratio of $1 + (\frac{m-1}{m})\frac{\alpha^2}{2}$ and $\frac{k\alpha^2}{\alpha^2+k-1}\left[1 + \frac{k-1}{m}\right] + \frac{m-k}{m}$ respectively.

IV. Model 1: No Replication

In this problem model we have considered situation where each task is restricted to be scheduled on only one machine, i.e. $|M_j| = 1$. We have a set J of n jobs, and a set M of m machines. 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 construction to phase one only. There is no replication of tasks in this model.

A. Lower Bound

Theorem IV.1. *For the model having $|M_j| = 1$, there is no online algorithm having competitive ratio better than $\frac{\alpha^2 m}{\alpha^2+m-1}$.*

Proof: We use adversary lower bound technique to prove the theorem. We define a lower bound input instance for an unknown correct algorithm. The adversary maximizes the competitive ratio until it becomes feasible to be scheduled by any algorithm.

Figure 1 shows the scenario where $|M_j| = 1$. Let us consider an instance in which total tasks be λm with $\tilde{p_i} = 1$. After scheduling let the most loaded machine j have B tasks. Obviously, $B \geq \lambda$. In phase 2 the adversary increases the tasks on j by α and decreases the other tasks by $\frac{1}{\alpha}$. So, C_{max} = αB and $C^*_{max} \geq \frac{\alpha B + \frac{1}{\alpha}(\lambda m - B)}{m}$. We have,

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

From above expression it is clear that smaller the value of B, the value of $\frac{C_{max}}{C^*_{max}}$ decreases. So, any algorithm tries to minimize B to get better performance. But for a schedule to become feasible the condition, $B \geq \lambda$ must be satisfied. Irrespective of the value of λ, for $B \geq \lambda$ the value of expression would be feasible minimum and would equal to $\frac{\alpha^2 m}{\alpha^2+m-1}$. ■

Corollary IV.1.1. *When $m \to \infty$ there is no online algorithm having competitive ratio better than α^2.*

B. Algorithm

We introduce an algorithm, **LPT-No Choice** which is based on Graham's LPT algorithm. In phase 1, The algorithm schedules the jobs to different machines in offline mode based on non-increasing order of their estimated processing times. Just like Graham's LPT a job is assigned to a machine having least load at that time. Adhering to the construction of the model, there is no choice of load balancing based on replication in phase 2. The performance of the algorithm depends on how much actual processing times of jobs vary from estimated ones. The actual processing time of a job may increase or shrink significantly from its estimated value and is known exactly only after the job is processed by a machine. So, the algorithm's performance incur the cost of estimation.

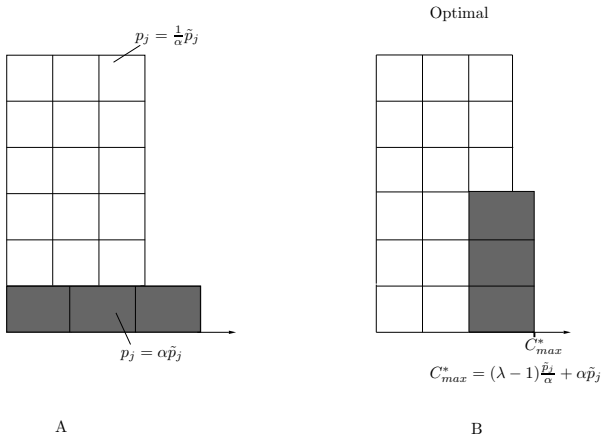


Figure 1: The figure represents the model having $|M_j| = 1$ with $\lambda = 3$ and $m = 6$. A represents worst case scenario showing all the tasks increased to α times in the most loaded machine. B represents optimal scheduling scenario.

Theorem IV.2. *The **LPT-No Choice** has a competitive ratio of* $\frac{2\alpha^2 m}{2\alpha^2+m-1}$.

Proof: The algorithm assigns the jobs to processors based on their estimated processing times using LPT in offline mode. So, the planned makespan considering estimated processing times of tasks, $\tilde{C}_{max}$ have following inequality relation with total estimated processing time, $\tilde{p_j}$ and estimated processing time of last task, l.

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

The actual makespan of a schedule, C_{max}, calculated after actual processing times of all the jobs are known,must be smaller than $\alpha\tilde{C}_{max}$. Considering this we have following inequality:

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

Considering the worst case situation the processor with $\tilde{C}_{max}$ will see its tasks increase by α and the load on the rest of the processors will by shrink $\frac{1}{\alpha}$. The argument behind this statement is that greater the value of ratio $\frac{C_{max}}{\sum p_j}$, the algorithm will give bad approximation. So increasing the load on the machine which have C_{max} and deceasing the rest of the load on other processors will reach worst case scenario. So the total actual processing time is given by the following equation.

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

Also the actual optimal makespan have following constraint

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

Substituting for $\sum p_j$, we have

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

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

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

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

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

The Graham's LPT algorithm in normal scenario give optimal performance till two jobs are assigned to machines. But in the scenario of estimated processing times of tasks whose value can significantly vary from actual actual ones, any schedule using estimated values cannot guarantee optimality for even two jobs per machine. So, we consider the instance of at least two jobs per machine to derive performance ratio of our algorithm. For at least two jobs in machine having $\tilde{C}_(max)$, the processing time (estimated) of latest job, $\tilde{p}_l$ at most be equal to $\tilde{C}_(max)/2$. Putting this in the above equation, we have

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

Using equation 3,

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

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

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

■

V. Model 2: replication is done everywhere

In this model, we consider putting no restriction on tasks assignment. The tasks are allowed to be replicated everywhere i.e. $\forall j, |M_j| = |M|$. We introduce **LPT-No Restriction** algorithm which is also a two phase algorithm. In the first phase no restriction is put into task assignment. It is assumed that each task can be replicated on any machine. In the second phase we simply use the Longest Processing Time algorithm (LPT) using estimated processing times of tasks in non-increasing order as input.

Lemma V.1. ***LPT-No Restriction*** *has a makespan of at least $\frac{2}{\alpha^2}p_l$ when there are at least two tasks on the machine to which the last task l is scheduled.*

Proof: 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

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

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

■

Theorem V.2. ***LPT-No Restriction*** *has a competitive ratio of $\frac{C_{max}}{C^*_{max}} \leq 1 + (\frac{m-1}{m})\frac{\alpha^2}{2}$*

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

$$C^*_{max} \geq \frac{\sum p_j}{m} \tag{5}$$

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

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

$$C_{max} \leq \frac{\sum p_j}{m} + \frac{(m-1)}{m}p_l \tag{6}$$

Using 5 and 6, we have

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

Using Lemma V.1, we have

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

■

Graham's List Scheduling algorithm always has approximation of $2 - \frac{1}{m}$. For $\alpha^2 < 2$, the **LPT- No Restriction** algorithm has better approximation than LS algorithm. For $\alpha^2 > 2$ the LS algorithm has better guarantee because $2p_l/\alpha^2$ becomes smaller p_l. As **LPT-No Restriction** is a variant of LS algorithm the optimal makespan must be at least p_l. So $C^*_{max} \geq max[2p_l/\alpha^2, p_l]$ holds always true and hence the algorithm has approximation ratio of $max[1 + ((m-1)/2m)\alpha^2, 2 - 1/m]$.

VI. Model 3: Replication in groups

Figure 2 shows the construction of two phases in model 3. In this model 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 tasks. There are k groups $G1$,$G2$...Gk. Size of each group is equal and have m/k processors within each group. We have considered task allocation such that each task can be assigned to only one group, i.e. $\forall j$, $|M_j| = m/k$. So, this model allows each task to replicate m/k times within the group it is assigned to.

We propose a two-phase algorithm, **LS-Group** which is based on Graham's List Scheduling algorithm. In phase 1, using LS in offline mode the algorithm pre-assign the jobs to different groups of machines. In phase 2 each task is scheduled to a particular processor within the group it was allocated in phase 1. In phase 2, the algorithm use online LS to schedule tasks to processors within each group.

Theorem VI.1. *When the number of groups is k the approximation ratio of **LS-Group** is $\frac{k\alpha^2}{\alpha^2+k-1}[1+\frac{k-1}{m}]+\frac{m-k}{m}$*

Proof: Let us consider that C_{max} comes from group $G1$. As in phase 1 tasks are allocated to different groups using offline List Scheduling algorithm and taking the estimated processing times of tasks, the load difference between any two groups cannot be greater

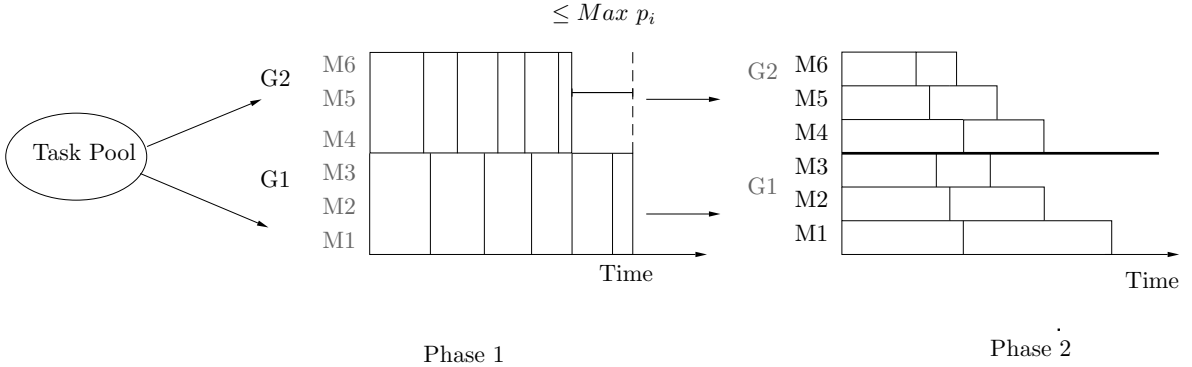


Figure 2: Replication in groups with $m = 6$, $k = 2$. The left figure shows tasks assignment to groups in phase 1; The right figure shows each task assigned to a machine within a group

than the estimated value of largest task $max_{i\in T}\tilde{p}_i$. 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

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

Case 1: When $(k-1)\sum_{i\in G1}\tilde{p}_i > \sum_{l=2}^{k}\sum_{i\in Gl}\tilde{p}_i$.

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

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

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

$$\sum_{l=2}^{k}\sum_{i\in Gl}p_i \geq (k-1)[\frac{1}{\alpha^2}\sum_{i\in G1}p_i - max_{i\in T}p_i] \quad (7)$$

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

$$C_{max} \leq \frac{\sum_{i\in G1}p_i}{m/k} + \frac{m/k-1}{m/k}p_{max} \quad (8)$$

Where p_{max} is actual processing time of largest task in $G1$.

Also, C^*_{max} must be greater than the average of the loads on the machines. $\sum_{i\in T}p_i$ represents sum of processing times all the tasks.

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

$\sum_{i\in T}p_i$ can be written as sum of load on $G1$ and load on rest of groups.

$$C^*_{max} \geq \frac{\sum_{i\in G1}p_i + \sum_{l=2}^{k}\sum_{i\in Gl}p_i}{m} \quad (9)$$

from 7, we derive

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

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

$$\frac{\alpha^2}{\alpha^2+k-1}(mC^*_{max} + (k-1)max_{i\in T}p_i) \geq \sum_{i\in G1}p_i \quad (10)$$

Using 8 and 10, We have

$$C_{max} \leq \frac{k\alpha^2}{\alpha^2+k-1}\left(C^*_{max} + \frac{(k-1)}{m}max_{i\in T}p_i\right) + \frac{m/k-1}{m/k}p_{max}$$

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

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

So, for Case 1 the algorithm has approximation ratio,

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

Case 2: When $(k-1)\sum_{i\in G1}\tilde{p}_i \leq \sum_{l=2}^{k}\sum_{i\in Gl}\tilde{p}_i$.

As processing times of tasks can change within a factor α and $\frac{1}{\alpha}$ of their estimated values, the expression for case 2 can be written as

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

Putting this value in equation 9, we have

$$C^*_{max} \geq \frac{\alpha^2+k-1}{m\alpha^2}\sum_{i\in G1} p_i \tag{11}$$

Using 8 and 11, and as $C^*_{max} \geq p_{max}$, we have

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

So, for case 2 the algorithm has approximation ratio of $\frac{k\alpha^2}{\alpha^2+k-1}+\frac{m-k}{m}$.
Clearly, the algorithm has worst approximation for case 1. So, the algorithm has approximation ratio of $\frac{C_{max}}{C^*_{max}} \leq \frac{k\alpha^2}{\alpha^2+k-1}\left[1+\frac{k-1}{m}\right]+\frac{m-k}{m}$

■

Corollary VI.1.1. *When the number of groups is* 2 *the approximation ratio is* $1+\frac{2}{1+\alpha^2}(\alpha^2-\frac{1}{m})$

LS-Group uses the LS algorithm in both its phases. A LPT-based algorithm may have better guarantee. But for no replication scenario i.e. when number of groups $k=m$, the **LS-Group** algorithm has an approximation ratio almost equal to **LPT-No choice's** when the number of machines m is large and the value of α is within practical range. This concludes that the algorithm has a guarantee almost equal to the one of LPT-based algorithm would have for most of the practical scenario.

VII. Summary

Table I summarizes the results of the three algorithms in terms of approximation ratio. Based on adversary technique, Theorem 1.1 states that there is no algorithm which can give performance better than α^2 for the model where no replication is allowed. LPT-No Choice is $\frac{2\alpha^2 m}{2\alpha^2+m-1}$ approximation in this model. For the second model having $|M_j| = |M|$ (replication everywhere), LPT-No Restriction has a competitive ratio of $1+(\frac{m-1}{m})\frac{\alpha^2}{2}$. The third model having $|M_j| = m/k$, allows replication within a group of m/k machines to which a task is assigned in phase 1. For this model, LS-Group algorithm has a competitive ratio of $\frac{k\alpha^2}{\alpha^2+k-1}\left[1+\frac{k-1}{m}\right]+\frac{m-k}{m}$.

To better understand the relation between these different results we show in figure 3 how the expression of the guarantees translate to actual values in a ratio- replication space. We picked 3 values of α while keeping the number of machines fixed $m = 210$. For replication everywhere scenario the number of replication is full and is always equal to total number of machines. For no replication scenario the approximation ratio is fixed for each value of α. For the model where replication is allowed within the group the approximation ratio decreases significantly with few replications. We observe that after a significant number of replications the ratio does not improve much further.

VIII. Conclusion and Future Work

We study the effect on performance of parallel machine scheduling when jobs are scheduled based on their estimated values of processing times. We define 3 different models which uses different replication schemes. We propose an approximation algorithm per model and compare their performance. We observed that even for small amount of replications (i.e for small group size), the guarantee of algorithm improves drastically. So, allowing a job to be replicated over large number of machines (large group size) might be unnecessary.

There are some open problems which can be explored as further research work. For models allowing more than one replica of a job, there is scope of introducing algorithms having better lower bounds.

In this research our focus was to study the effect of replication for minimizing makespan in scheduling under uncertainty. We did not look in the problem through memory point of view. Replication facilitates load balancing and allows better response time by the processors and but it is having memory cost attached with it. So, one direction to think about the problem would be solving bi criteria problem of optimizing load as well as memory.

Algorithms that replicate task without making groups of processors could give better guarantees as there would be no restriction on the processors where a job can be replicated. To reduce memory cost non-constant or selective replication can be done: replicating big tasks only could reduce the cost of replication.

References

[1] J. Abawajy. Placement of file replicas in data grid environments. In M. Bubak, G. van Albada, P. Sloot,

Replication	Approximation ratio
$\lvert M_j\rvert = 1$	$\frac{C_{max}}{C^*_{max}} \leq \frac{2\alpha^2 m}{2\alpha^2+m-1}$ [Th. 1.2] No approximation better than α^2 [Th. 1.1]
$\lvert M_j\rvert = \lvert M\rvert$	$\frac{C_{max}}{C^*_{max}} \leq 1 + (\frac{m-1}{m})\frac{\alpha^2}{2}$ [Th. 2.1] $2 - \frac{1}{m}$ [Graham's LS]
$\lvert M_j\rvert = m/k$	$\frac{C_{max}}{C^*_{max}} \leq \frac{k\alpha^2}{\alpha^2+k-1}\left[1 + \frac{k-1}{m}\right] + \frac{m-k}{m}$ [Th. 3.1] $\frac{C_{max}}{C^*_{max}} \leq 1 + \frac{2}{1+\alpha^2}\left(\alpha^2 - \frac{1}{m}\right)$ when $k = 2$ [Col. 3.1]

Table I: Summary Table

and J. Dongarra, editors, *Computational Science - ICCS 2004*, volume 3038 of *Lecture Notes in Computer Science*, pages 66–73. Springer Berlin Heidelberg, 2004.

[2] L.-C. Canon and E. Jeannot. Evaluation and Optimization of the Robustness of DAG Schedules in Heterogeneous Environments. 2010. Accepted for publication.

[3] V. Cardellini, R. I, M. Colajanni, and P. S. Yu. Dynamic load balancing on web-server systems. *IEEE Internet Computing*, 3:28–39, 1999.

[4] W. Cirne, F. Brasileiro, D. Paranhos, L. F. W. Góes, and W. Voorsluys. On the efficacy, efficiency and emergent behavior of task replication in large distributed systems. *Parallel Computing*, 33(3):213 – 234, 2007.

[5] R. L. Daniels and P. Kouvelis. Robust Scheduling to Hedge Against Processing Time Uncertainty in Single-Stage Production. *MANAGEMENT SCIENCE*, 41(2):363–376, Feb. 1995.

[6] A. J. Davenport, C. Gefflot, and J. C. Beck. Slack-based techniques for robust schedules.

[7] G. Erlebacher, E. Saule, N. Flyer, and E. Bollig. Acceleration of derivative calculations with application to radial basis function - finite-differences on the Intel MIC architecture. In *Proc. of International Conference on Supercomputing (ICS)*, 2014.

[8] M. Gatto and P. Widmayer. On the robustness of graham's algorithm for online scheduling. In *Proc of WADS*, 2007.

[9] R. L. Graham. Bounds on multiprocessing timing anomalies. *SIAM JOURNAL ON APPLIED MATHEMATICS*, 17(2):416–429, 1969.

[10] S. Kavulya, J. Tan, R. Gandhi, and P. Narasimhan. An analysis of traces from a production mapreduce cluster. In *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, CCGRID '10, pages 94–103, Washington, DC, USA, 2010. IEEE Computer Society.

[11] Z. Li and M. G. Ierapetritou. Process scheduling under uncertainty: Review and challenges. *Computers & Chemical Engineering*, 32(4-5):715–727, 2008.

[12] R. Lipton and J. Naughton. Query size estimation by adaptive sampling. *Journal of Computer and System Sciences*, 51(1):18 – 25, 1995.

[13] M. Pastorelli, A. Barbuzzi, D. Carra, M. Dell'Amico, and P. Michiardi. Hfsp: Size-based scheduling for hadoop. In *Big Data, 2013 IEEE International Conference on*, pages 51–59, Oct 2013.

[14] R. Rahman, K. Barker, and R. Alhajj. Study of different replica placement and maintenance strategies in data grid. In *Cluster Computing and the Grid, 2007. CCGRID 2007. Seventh IEEE International Symposium on*, pages 171–178, May 2007.

[15] S. S. H. Tse. Online bounds on balancing two independent criteria with replication and reallocation. *IEEE Trans. Computers*, 61(11):1601–1610, 2012.

[16] A. Verma, L. Cherkasova, and R. H. Campbell. Aria: Automatic resource inference and allocation for mapreduce environments. In *Proceedings of the 8th ACM International Conference on Autonomic Computing*, ICAC '11, pages 235–244, New York, NY, USA, 2011. ACM.

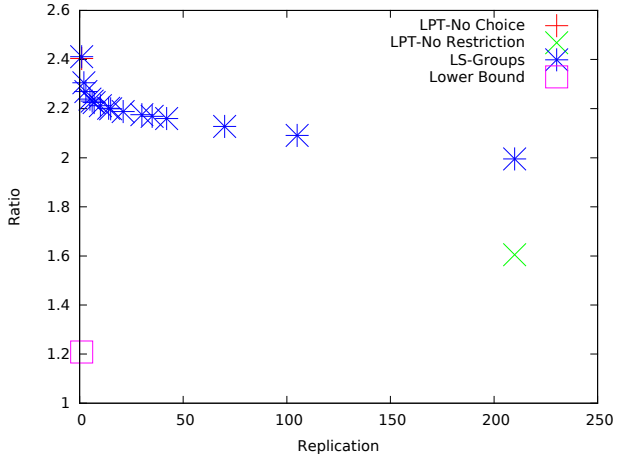
[17] D. Wang, G. Joshi, and G. W. Wornell. Efficient task replication for fast response times in parallel computation. *CoRR*, abs/1404.1328, 2014.

[18] T. White. *Hadoop: The Definitive Guide*. O'Reilly Media, Inc., 1st edition, 2009.

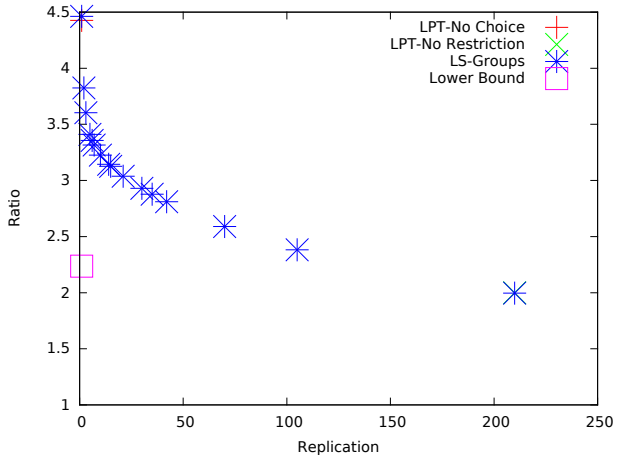
[19] A. Wierman and M. Nuyens. Scheduling despite inexact job-size information. In Z. Liu, V. Misra, and P. J. Shenoy, editors, *SIGMETRICS*, pages 25–36. ACM, 2008.

[20] J. Wolf, D. Rajan, K. Hildrum, R. Khandekar, V. Kumar, S. Parekh, K.-L. Wu, and A. balmin. Flex: A slot allocation scheduling optimizer for mapreduce workloads. In *Proceedings of the ACM/IFIP/USENIX 11th International Conference on Middleware*, Middleware '10, pages 1–20, Berlin, Heidelberg, 2010. Springer-Verlag.

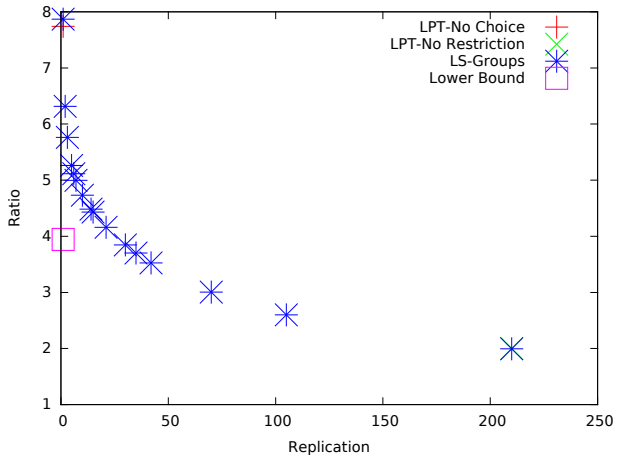
[21] Y. You, D. A. Bader, and M. M. Dehnavi. Designing a heuristic cross-architecture combination for breadth-first search. In *Proc. of the 43rd International Conference on Parallel Processing*, 2014.



(a) $m = 210$, $\alpha = 1.1$



(b) $m = 210$, $\alpha = 1.5$



(c) $m = 210$, $\alpha = 2$

Figure 3: Ratio-Replication graph with m=210 and value of α as 1.1, 1.5 and 2