Report Number: WUCSE-2012-24

2012

Correction of an Augmentation Bound Analysis for Parallel Real-Time Tasks

Abusayeed Saifullah, Kunal Agrawal, Chenyang Lu, and Christopher Gill

This paper proposes some significant corrections in a recent work of Lakshmanan et al on parallel task scheduling. Lakshmanan et al have proposed a transformation of parallel tasks into sequential tasks, and have claimed a resource augmentation bound of 3:42 for partitioned deadline monotonic (DM) scheduling of the transformed tasks. We demonstrate that their analysis for resource augmentation bound is incorrect. We propose a different technique for task transformation that requires a resource augmentation bound of 5 for partitioned DM scheduling.

2012-24

Correction of an Augmentation Bound Analysis for Parallel Real-Time Tasks

Authors: Abusayeed Saifullah, Kunal Agrawal, Chenyang Lu, Christopher Gill

Abstract: This paper proposes some significant corrections in a recent work of Lakshmanan et al on parallel task scheduling. Lakshmanan et al have proposed a transformation of parallel tasks into sequential tasks, and have claimed a resource augmentation bound of 3:42 for partitioned deadline monotonic (DM) scheduling of the transformed tasks. We demonstrate that their analysis for resource augmentation bound is incorrect. We propose a different technique for task transformation that requires a resource augmentation bound of 5 for partitioned DM scheduling.

Type of Report: Other

Noname manuscript No.
(will be inserted by the editor)

Correction of an Augmentation Bound Analysis for Parallel Real-Time Tasks

Abusayeed Saifullah · Kunal Agrawal · Chenyang Lu · Christopher Gill

Abstract This paper proposes some significant corrections in a recent work of Lakshmanan et al on parallel task scheduling. Lakshmanan et al have proposed a transformation of parallel tasks into sequential tasks, and have claimed a resource augmentation bound of 3.42 for partitioned deadline monotonic (DM) scheduling of the transformed tasks. We demonstrate that their analysis for resource augmentation bound is incorrect. We propose a different technique for task transformation that requires a resource augmentation bound of 5 for partitioned DM scheduling.

1 Introduction

With the advent of multi-core processors, real-time scheduling of parallel tasks has received considerable attention in recent years. In this paper, we propose some significant corrections in a recent work of Lakshmanan et al (2010) on real-time scheduling of parallel tasks on multi-core processors.

Task Model by Lakshmanan et al. Lakshmanan et al (2010) have addressed real-time scheduling of a restricted synchronous parallel task model (which they call Fork-Join model) where each parallel task is an alternate sequence of parallel and sequential segments. All parallel segments have an equal number of parallel threads, and the execution requirements of all threads in any segment are equal. Also, the number of threads in every segment is no greater than the total number of processor cores. An example of their task model is shown in Figure 1, where the horizontal bars indicate the lengths of execution requirements of the threads. We call this model *synchronous*, since all the threads of a parallel segment must finish before the next segment starts, creating a synchronization point shown as vertical lines in Figure 1. A parallel task, denoted by τ_i, has s_i segments. The execution requirement of each thread in the j-th segment, $1 \leq j \leq s_i$ is denoted by $e_{i,j}$. The total number of threads in the j-th segment, $1 \leq j \leq s_i$ is denoted by $m_{i,j}$. The number of processor cores is denoted by m. For every task τ_i, $m_{i,j} \leq m$, and for any two parallel segments j and k of τ_i, $m_{i,j} = m_{i,k}$. The period of τ_i is denoted by T_i. The deadline of τ_i is equal to T_i. For any task τ_i, its maximum execution time $C_i = \sum_{j=1}^{s_i} m_{i,j} e_{i,j}$, minimum execution time $\eta_i = \sum_{j=1}^{s_i} e_{i,j}$, and utilization $u_i = \frac{C_i}{T_i}$. For a set of n tasks, the total utilization $u_{\text{sum}} = \sum_{i=1}^{n} u_i$.

Results by Lakshmanan et al. Lakshmanan et al (2010) have proposed a task transformation that converts each parallel task into a set of smaller sequential tasks, called *subtasks*. Each task τ_i with $C_i > T_i$ is stretched up to its deadline T_i to create a *master subtask* with execution requirement equal to T_i. Thus each task

Abusayeed Saifullah, Kunal Agrawal, Chenyang Lu, Christopher Gill
Computer Science and Engineering, Washington University, St Louis, MO 63130

τ_i with $C_i > T_i$ is *fully stretched*, and converted to a master subtask with execution requirement equal to T_i, and a set of constrained deadline subtasks. Each task τ_j with $C_j \leq T_j$ cannot be fully stretched, and is converted to one sequential subtask. The precedence constraints are retained by assigning each subtask a (sub)deadline and a release offset. For partitioned deadline monotonic (DM) scheduling, Lakshmanan et al (2010) have proposed *Partitioned FJ-DMS* (Fork-Join Deadline Monotonic Scheduling) algorithm for the transformed tasks. In Partitioned FJ-DMS, each master subtask that is fully stretched up to the task deadline is exclusively assigned one processor core. All other subtasks are partitioned based on the FBB-FFD (Fisher Baruah Baker - First Fit Decreasing) algorithm (Fisher et al, 2006). Lakshmanan et al (2010) claim that if the original set of parallel tasks is schedulable on an m-core unit-speed machine, then the transformed subtasks are guaranteed to be scheduled under the Partitioned FJ-DMS algorithm on m processor cores each having a speed of 3.42.

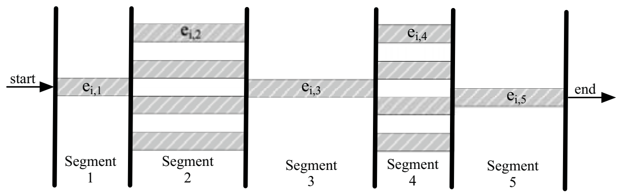


Fig. 1 A parallel task τ_i

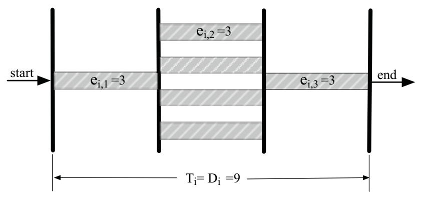
Our contributions. In this paper, we demonstrate that the analysis provided by Lakshmanan et al (2010) for proving the resource augmentation bound is incorrect. Since Lakshmanan et al (2010) do not specify whether the task transformation is performed on a unit-speed machine or on a 3.42 (or above) speed machine, we present our arguments from both points of view. First show that the resource augmentation bound does not hold if the task transformation is done on a unit-speed machine, and the transformed (sub)tasks are scheduled on a 3.42-speed machine. Then we consider the task transformation on a ν-speed machine ($\nu \geq 3.42$), and show that the analysis used for proving the resource augmentation bound is incorrect. There seems to be no easy way of correcting the existing analysis, and hence we propose a different technique for task transformation that requires a resource augmentation bound of 5 for partitioned DM scheduling.

2 Transformation on Unit-speed Processor

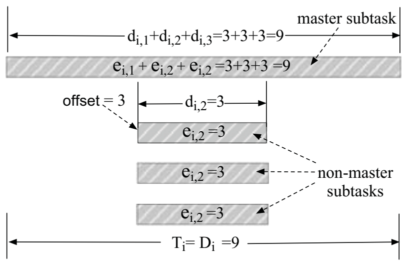
In this section, we refute the analysis and resource augmentation bound for the case where the tasks are transformed on unit-speed processors and the transformed tasks are scheduled on 3.42-speed processors.

2.1 A Counter Example

Lakshmanan et al (2010) claim that if a set of parallel tasks is feasible on m processor cores each of unit-speed, then the transformed tasks can be partitioned



(a) A parallel task τ_i



(b) Transformed subtasks of τ_i

Fig. 2 A counter example

under the Partitioned FJ-DMS algorithm on an m-core machine where each core has speed 3.42. To prove this, Theorem 9 in Lakshmanan et al (2010) proves that for any task set with $u_{\text{sum}} \leq m$, and $\eta_i \leq T_i$ for each task τ_i, the Partitioned FJ-DMS algorithm is guaranteed to partition the transformed tasks on m processor cores each of speed 3.42. As follows, we refute the results by showing a counter example with a task set with $u_{\text{sum}} \leq m$, and $\eta_i \leq T_i$ for each task τ_i, where the transformed tasks cannot be partitioned under the Partitioned FJ-DMS algorithm.

Let us consider 2 tasks, each having the structure as shown in Figure 2(a). Each thread has an execution requirement of 3. For each task τ_i, $C_i = 18$, $\eta_i = 9$, and $T_i = D_i = 9$. Let us consider the total number of processor cores $m = 4$. Note that for these 2 tasks, $u_{\text{sum}} = \frac{18}{9} + \frac{18}{9} = 4 \leq m$. In the transformation of task τ_i, Lakshmanan et al (2010) create a master subtask with execution requirement (and also deadline) equal to T_i. All other threads are converted to non-master subtasks with constrained deadlines. Let the deadline assigned to each non-master subtask of the j-th segment be denoted by $d_{i,j}$. The transformed subtasks of one task is shown in Figure 2(b). Upon transformation, for each task, one master subtask is created with deadline =execution requirement=9 by taking one thread from each segment. The remaining 3 threads on the 2nd segment are converted to 3 subtasks, each with a deadline of 3. Thus, for 2 such parallel tasks, we will have a total of 2 master subtasks (each with deadline =execution requirement=9) and 6 non-master subtasks (each with deadline =execution requirement=3) upon transformation. According to the Partitioned FJ-DMS algorithm (Lakshmanan et al, 2010), each master subtask is assigned one core exclusively. The non-master subtasks are assigned according to the FBB-FFD algorithm. But for this example task set, it can be easily seen that once we assign exclusively one core to each master subtask, the remaining subtasks cannot be partitioned using the FBB-FFD algorithm when each core has speed 3.42.

2.2 The Oversights in Existing Analysis

Lakshmanan et al (2010) have analyzed the resource augmentation bound based on the demand bound function (DBF) and load of the transformed subtasks. For a task τ_i with a maximum computation requirement of C_i, a period of T_i, and a deadline of D_i, its DBF in an interval of t time units is given by $\text{DBF}(\tau_i, t) = \max\left(0, \left(\lfloor \frac{t-D_i}{T_i} \rfloor + 1\right)C_i\right)$. The *load* of a task set τ, denoted by $\delta_{\text{sum}}(\tau)$, is defined as $\delta_{\text{sum}}(\tau) = \max_{t>0}\left(\frac{\sum_{i=1}^{n} \text{DBF}(\tau_i,t)}{t}\right)$. For n parallel tasks, Lakshmanan et al (2010) have proved that the load δ_{sum} of all subtasks on unit-speed processor cores is

$$\delta_{\text{sum}} \leq \sum_{i=1}^{n} \frac{C_i}{T_i - \eta_i} \tag{1}$$

From Condition (1), the load $\delta_{\text{sum},\nu}$ on ν-speed cores has been claimed to be

$$\delta_{\text{sum},\nu} \leq \sum_{i=1}^{n} \frac{C_i/\nu}{T_i - \eta_i/\nu} \tag{2}$$

Oversights in load calculation. We demonstrate that Condition (2) is *incorrect*, and so is the augmentation bound. This can be easily demonstrated using the same example of 2 tasks used in the previous section. For example, when $\nu = 3$, the load $\delta_{\text{sum},3}$ for the subtasks of these 2 tasks according to Condition (2) becomes

$$\delta_{\text{sum},3} \leq \frac{18/3}{9 - 9/3} + \frac{18/3}{9 - 9/3} = 2 \tag{3}$$

Now, in Figure 2(b) for 1 parallel task, we can see that upon transformation of 2 parallel tasks of our example, $\delta_{\text{sum},3} > 2$, thereby violating Condition (3).

Correction of load. From Condition (1), the correct bound of $\delta_{\text{sum},\nu}$ should be

$$\delta_{\text{sum},\nu} = \frac{\delta_{\text{sum}}}{\nu} \leq \frac{1}{\nu}\sum_{i=1}^{n}\frac{C_i}{T_i - \eta_i} \tag{4}$$

If we use the correct upper bound of $\delta_{\text{sum},\nu}$, in the analysis of Lakshmanan et al (2010), the results end up with a denominator 0, leading to an undefined condition.

Precedence violations. The stretch transformation does not preserve the precedence relations in the original task. For example, in Figure 2(b), when each processor core is of speed 3, the master subtask can finish in first 3 time units, while the non-master subtasks have not yet started due to offset, thereby violating the precedence constraints in the original task shown in Figure 2(a).

3 Transformation on ν-speed Processor

Now we present our arguments by considering task transformation on ν ($\nu \geq 3.42$) speed processors. We show that if Lakshmanan et al (2010) perform task transformation on ν ($\nu \geq 3.42$) speed processors, then their current proof is incorrect.

3.1 Correction of the Partitioned FJ-DMS Algorithm

According to the Partitioned FJ-DMS algorithm, a master subtask is assigned one processor core exclusively if the (original) task can be fully stretched on ν ($\nu \geq 3.42$) processor cores. That is, if a task cannot be stretched fully on ν ($\nu \geq 3.42$) processor cores, then it is partitioned using the FBB-FFD algorithm. In the analysis for the Partitioned FJ-DMS algorithm, Lakshmanan et al (2010) use δ_{max} to denote the *maximum density* among all subtasks that are partitioned under the FBB-FFD algorithm, where the *density* of a task or subtask is defined as the ratio of its maximum execution requirement to its deadline (i.e., C_i/D_i). In the proof of Theorem 9, Lakshmanan et al (2010) claim that δ_{max} is at most $\frac{1}{\nu}$. We notice that a task that cannot be stretched fully on ν ($\nu \geq 3.42$) processor cores can have execution requirement very close to but smaller than its deadline on ν ($\nu \geq 3.42$) processor cores. Thus, such a (sub)task that is partitioned under the FBB-FFD algorithm can have *density* very close to 1, thereby violating the bound of $\frac{1}{\nu}$. For example, a task with period 100 and a maximum execution requirement of 99 on 3.42 speed cores will have a density of 0.99 on 3.42 speed cores. Note that using $\delta_{max} = 1$ in the current analysis results, in denominator near to 0, providing an undefined condition or no practical resource augmentation bound.

If we adhere to the current analysis approach of Lakshmanan et al (2010), then the Partitioned FJ-DMS algorithm should be modified as follows. Any master task that has density larger than $\frac{1}{\nu}$ on ν-speed ($\nu \geq 3.42$) processor cores should be assigned one processor core exclusively. This will ensure that for the (sub)tasks that are partitioned based on the FBB-FFD algorithm, δ_{max} is at most $\frac{1}{\nu}$.

3.2 Oversights in the Analysis

We now try to follow the proof of Theorem 9 of Lakshmanan et al (2010). In the proof, they have used a sufficient condition for partitionability of the tasks under the FBB-FFD algorithm. For each parallel task, if there is a master subtask of density larger than $\frac{1}{\nu}$ on ν-speed ($\nu \geq 3.42$) cores, then the Partitioned FJ-DMS algorithm assigns one core exclusively to the master subtask. The remaining processor cores are used to partition the remaining subtasks using the FBB-FFD algorithm. But Lakshmanan et al (2010) prove a resource augmentation bound 3.42 by considering that all m processors are available for the (sub)tasks that are partitioned using the FBB-FFD algorithm. That is, even though some processor cores are exclusively assigned to some master tasks, these cores are again considered for partitioning other subtasks under the FBB-FFD algorithm. Therefore, the proof for resource augmentation bound of 3.42 is not correct. Even if we take the above correction into consideration, then it can be seen that there is no easy way of proving a resource augmentation bound by leveraging the current analysis approach of Lakshmanan et al (2010).

4 Correction of Task Transformation and Analysis

In the previous sections, we have seen that the current result of Lakshmanan et al (2010) does not hold or cannot be proven correctly from any point of view. In a previous work (Saifullah et al, 2011), we have derived a resource augmentation bound of 5 under the FBB-FFD algorithm using a different task transformation for a more general parallel task model. Hence, those results in Saifullah et al (2011) still hold for the restricted task model of Lakshmanan et al (2010).

References

Fisher N, Baruah S, Baker TP (2006) The partitioned scheduling of sporadic tasks according to static-priorities. In: ECRTS ’06, pp 118–127

Lakshmanan K, Kato S, Rajkumar RR (2010) Scheduling parallel real-time tasks on multi-core processors. In: RTSS ’10, pp 259–268

Saifullah A, Agrawal K, Lu C, Gill C (2011) Multi-core real-time scheduling for generalized parallel task models. In: RTSS ’11, pp 217–226