Performance Analysis for Segment Stretch Transformation of Parallel Real-time Tasks

Manar Qamhieh, Frédéric Fauberteau, Serge Midonnet

Performance Analysis for Segment Stretch Transformation of Parallel Real-time Tasks

Manar Qamhieh, Frédéric Fauberteau, Serge Midonnet
Université Paris-Est
LIGM, UMR CNRS 8049
5 bd Descartes 77454 Marne la vallée CEDEX 2
{manar.qamhieh,frederic.fauberteau,serge.midonnet}@univ-paris-est.fr

***Abstract*—The Segment Stretch Transformation (SST) is an algorithm that transforms parallel Fork-Join (FJ) tasks into sequential tasks on multiprocessor systems when possible, in order to increase the schedulability of the tasksets of this model. SST is based on Task Stretch Transformation (TST) which is a transformation for the same model of tasks, but it uses segment migrations while SST eliminates their use.**

In this paper, we prove that SST transformation has the same performance of TST transformation by providing a detailed analysis based on Demand Bound Function (DBF) and by showing that SST has a resource augmentation bound of 3.42, same as TST, which means that if a taskset is feasible on m speed processors, then it is schedulable using the transformation on m processors that are 3.42 times faster.

I. Introduction

Parallelism in normal systems is widely used for a long time, and it is created specially to cope with the tendency of chip manufacturers to build multiprocessor systems. But in real-time systems, the integration of parallelism is more complicated when it is to be compared with ordinary sequential tasks even if they are executed on multiprocessor systems. Parallelism in real-time systems can be defined as the execution of the same task at the same time on multiple processors while respecting certain time constraints like period or deadline.

Parallelism has many models and theories that are applied in actual programming languages, like the MapReduce model which is designed by Google in order to speed up the execution of massive data on multiple processors. Another model is the fork-join model "FJ", which is a common parallel computing model and it is the base of OpenMP parallel programming C library [1].

The remainder of this paper is organized as follows: in Section II, we present our task model. Section III describes the transformations TST and SST. Section IV contains the analysis of the demand bound function and resource augmentation bound done on the SST, and we finish with the conclusion in section V.

II. Fork-Join Model

The parallel real-time task of fork-join model is a task in which certain parts are executed simultaneously on multiple processors. As shown in Figure 1, FJ task consists of segments, both sequential and parallel. The task always starts by a sequential segment which is executed on one processor, and then it forks into a specific number of parallel segments, which they join together into a sequential segment, and so on. The number of segments are defined by the model, as well as the number of parallel processors, which means that all the parallel regions in the task share the same number of processors.

An implicit-deadline FJ task is described as the following: $\tau_i = ((C_i^1, P_i^2, C_i^3, \ldots, P_i^{s_i-1}, C_i^{s_i}), m_i, T_i)$ where:

- s_i is the total number of segments (sequential and parallel) and it is an odd number according to definition of the model,
- m_i is the number of parallel segments in each parallel region fixed by the model. $m_i > 1$ for parallel segments, and equal to 1 for sequential segments.
- C_i^k is the Worst Case Execution Time (WCET) of the k^{th} sequential segment, and k is an odd number and $1 \leq k \leq s_i$,
- P_i^k is the WCET of the parallel segments in the k^{th} parallel region, where k is an even number and $1 \leq k \leq s_i$, and $P_i^k = P_i^{k,1} = P_i^{k,2} = \ldots = P_i^{k,m_i}$
- T_i is the period of the task which is equal to the deadline D_i.

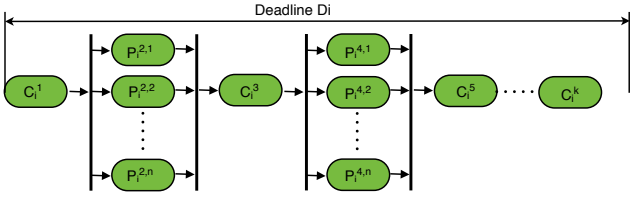

Fig. 1. Task of FJ model.

Definition II.1 (Master string)**.** *The master string of a parallel FJ task is a collection of segments that executes within the master thread of the task, starting by the first sequential segment and then the intermediate parallel and sequential segments, and ending by the last sequential segment. In numerical notations, master string can be represented as:*

$$\tau_i^1, \tau_i^{2,1}, \tau_i^3, \ldots, \tau_i^{(s_i-1),1}, \tau_i^{s_i}$$

Definition II.2 (Parallel execution length)**.** *The parallel execution length P_i is the sum of worst execution time of parallel*

segments in the master string of τ_i, where:

$$P_i = \sum_{k=1}^{\frac{s_i-1}{2}} P_i^{2k} \tag{1}$$

Definition II.3 (Minimum execution length)**.** *The minimum execution length η_i represents the minimum time a FJ task τ_i needs to execute when all parallel segments are executed in parallel. It is equal to the sum of WCET of all segments in the master string of task τ_i:*

$$\eta_i = (\sum_{k=0}^{\frac{s_i-1}{2}} (C_i^{2k+1})) + P_i \tag{2}$$

Definition II.4 (Maximum execution length)**.** *The maximum execution length C_i, which is the sum of WCET of all sequential and parallel segments in task τ_i:*

$$C_i = (\sum_{k=0}^{\frac{s_i-1}{2}} (C_i^{2k+1})) + m_i * P_i \tag{3}$$

Definition II.5 (Slack time)**.** *The slack time L_i is the temporal difference between the deadline and the minimum execution time.*

$$L_i = D_i - \eta_i \tag{4}$$

Definition II.6 (Capacity)**.** *The capacity f_i is defined as the capacity of the master string to execute parallel segments from all parallel regions within itself without missing its deadline.*

$$f_i = \frac{L_i}{P_i} \tag{5}$$

III. Related work

A. Task Stretch Transformation

According to Lakshmanan *et al.* in [2], the parallel real-time tasks of FJ model on multiprocessor systems can have schedulable utilization bound slightly greater than and arbitrarily close to uniprocessor schedulable utilization bound. Therefore they proposed an algorithm called TST.

The main objective of TST is to convert the parallel FJ task into sequential when possible, by creating a fully stretched master string where its execution time is equal to its deadline. Part of the parallel segments execute within the master string and the remaining ones are scheduled by a specific partitioned scheduling algorithm called FBB-FFD ({Fisher,Baruah,Baker}–First-Fit Decreasing) [3].

TST is proved to have a resource augmentation bound of 3.42, which means any taskset that is feasible on m unit speed processors is feasible using the TST on m processors of speed 3.42.

B. Segment Stretch Transformation

Since segment migrations are used heavily in the TST which limits its implementation by using a special Linux kernel called Linux/RK which supports semi-partitioning, we proposed earlier a modification to the algorithm called SST for the same FJ model of parallelism [4].

SST also tries to convert the parallel tasks into sequential ones by creating a master string, which in this case can be either fully stretched or not, in order to provide a partitioned scheduling for the parallel FJ tasks by eliminating the segment migration. As a result the new transformation algorithm can be implemented directly on a RT Linux kernel with no specific patches.

For a parallel implicit deadline task τ_i of FJ model, with maximum execution time of C_i, deadline of D_i, SST has 3 scenarios:

1) $C_i \leq D_i$: the task will be fully converted from parallel task to sequential,
2) $C_i > D_i$ and the master string will be fully stretched. The slack L_i of the master string will be completely filled by parallel segments and its execution time will be equal to the deadline,
3) $C_i > D_i$ and the master string will not be fully stretched, some of the slack time will remain unfilled.

In this paper we prove that SST have the same resource augmentation bound as TST, by providing a performance analysis on these 3 scenarios which is inspired by the analysis performed on TST algorithm in [2].

IV. Analysis

A. Demand Bound Function "DBF"

Definition IV.1 (DBF [5])**.** *DBF is defined as the largest cumulative execution requirement of all jobs that can be generated by τ_i to have both their arrival times and their deadlines within a contiguous interval of length t.*

For a task τ_i with a total execution time of C_i, period of T_i and a deadline of $D_i \leq T_i$ DBF is given by:

$$DBF(\tau_i, t) = max(0, (\lfloor \frac{t - D_i}{T_i} \rfloor + 1) C_i) \tag{6}$$

Theorem IV.1 (DBF)**.** *DBF of a stretched task $\tau_i^{stretched}$ using SST is:*

$$DBF(\tau_i^{stretched}, t) \leq \frac{C_i}{T_i - \eta_i} t$$

where $0 \leq \eta_i \leq T_i$.

Proof: In order to generalize DBF for the stretched task, the three cases of SST (Section III-B) have to be analyzed.

1) For the first case, the parallel task is transformed totally into a sequential one which is the master string, and $D_i^{master} = T_i^{master}$ and $C_i^{master} = C_i$ (Equation (3)). DBF is calculated as the following:

$$DBF(\tau_i^{stretched}, t) = DBF(\tau_i^{master}, t)$$

$$DBF(\tau_i^{stretched}, t) = \max\left(0, \left(\left\lfloor \frac{t - D_i}{T_i} \right\rfloor + 1\right) C_i^{master}\right)$$

$$DBF(\tau_i^{stretched}, t) = \max\left(0, \left(\left\lfloor \frac{t}{T_i} \right\rfloor\right) C_i\right) \leq \frac{C_i}{T_i} t$$

$$DBF(\tau_i^{stretched}, t) \leq \frac{C_i}{T_i - \eta_i} t$$

where $0 \leq \eta_i \leq T_i$.

2) The second case is when the master string is fully stretched but there exist parallel constrained deadline segments that are not part of the master string and will be scheduled using FBB-FFD:

$$\tau^{stretched} = \tau^{master} + \{\tau^{cd}\}$$

DBF will be as the following:

$$DBF(\tau_i^{stretched}, t) \leq DBF(\tau_i^{master}, t) + DBF(\{\tau_i^{cd}\}, t)$$

DBF for the master string can be calculated knowing that the master string is fully stretched, and $C_i^{master} = D_i^{master}$:

$$DBF(\tau_i^{master}, t) \leq \frac{C_i^{master}}{D_i^{master}} t$$
$$DBF(\tau_i^{master}, t) \leq t$$

The group of segments $\{\tau_i^{cd}\}$ consists of segments from all the parallel regions in τ_i, and only one parallel region is activated at time instant t. The maximum number of parallel segments in each region is $(q_i - 1)$, where $q_i = m_i - \lfloor f_i \rfloor$. Figure 2 shows an example of a stretched task of the second case, with constrained-deadline parallel segments of 3 different parallel regions.
Therefore, DBF can be calculated as the following:

$$DBF(\{\tau_i^{cd}\}, t) \leq \delta_i^{max}(q_i - 1)t \tag{7}$$

The density of a constrained deadline task τ_i is given by:

$$\delta_i = \frac{C_i}{D_i}$$

As shown also in Figure 2, the SST algorithm starts by filling the slack of the master string by $\lfloor f_i \rfloor$-parallel segments from each parallel region, and then we add other parallel segments if their WCET fits in the remaining slack. For a k parallel region, all τ^{cd} have the same WCET P_i^k, and deadline D_i^k where $D_i^k = (1 + n\lfloor f_i \rfloor)P_i^k$, and $1 \leq n < m_i$ according to the number of parallel segments from region which execute within the master string.
The maximum density of the parallel constrained deadline tasks τ^{cd} can be calculated as the following:

$$\delta_{max} = \max_{k=1}^{\frac{s_i - 1}{2}} \frac{P_i^{2k}}{(1 + n\lfloor f_i \rfloor)P_i^{2k}}$$
$$\delta_{max} = \max_{k=1}^{\frac{s_i - 1}{2}} \frac{1}{(1 + n\lfloor f_i \rfloor)}$$
$$\delta_{max} = \frac{1}{1 + \lfloor f_i \rfloor}$$

Since there exist at least one parallel region where $n = 1$, this region is the one with the highest density.
Using this result in Equation (7), the DBF is as the following:

$$DBF(\{\tau_i^{cd}\}, t) \leq \frac{1}{1 + \lfloor f_i \rfloor}(q_i - 1)t$$

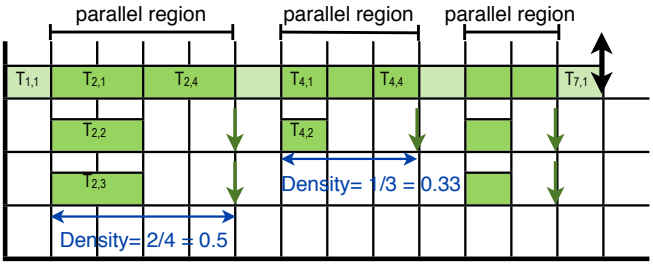

Fig. 2. SST: Parallel regions and constrained-deadline tasks $\{\tau^{cd}\}$.

In order to eliminate the use of $\lfloor f_i \rfloor$ in our calculations, we will use the following approximation:

$$\delta_i^{max} = \frac{1}{1 + \lfloor f_i \rfloor}$$
$$f_i - \lfloor f_i \rfloor < 1 \Longrightarrow 1 + \lfloor f_i \rfloor > f_i$$
$$\delta_i^{max} < \frac{1}{f_i}$$

which will lead to the following:

$$DBF(\{\tau_i^{cd}\}, t) \leq \frac{1}{f_i}(q_i - 1)t$$
$$f_i = \frac{T_i - \eta_i}{P_i}$$
$$DBF(\{\tau_i^{cd}\}, t) \leq \frac{(q_i - 1)P_i}{T_i - \eta_i} t \tag{8}$$

DBF of the whole stretched task can be calculated as the sum of both the master string and the group of constrained deadline, as the following:

$$DBF(\tau_i^{stretched}, t) \leq DBF(\tau_i^{master}, t) + DBF(\{\tau_i^{cd}\}, t)$$
$$DBF(\tau_i^{stretched}, t) \leq t + \frac{(q_i - 1)P_i}{T_i - \eta_i} t$$
$$DBF(\tau_i^{stretched}, t) \leq \frac{T_i - \eta_i + (m_i - \lfloor f_i \rfloor - 1)}{T_i - \eta_i} t$$
$$DBF(\tau_i^{stretched}, t) \leq \frac{m_i P_i}{T_i - \eta_i} t \leq \frac{C_i}{T_i - \eta_i} t$$

3) For the third case, the stretched task $\tau_i^{stretch}$ consists of a collection of constrained deadline tasks, including the master string, which will not be fully stretched.
The execution time of the master string in this case will be as the following:

$$\eta_i + P_i\lfloor f_i \rfloor \leq C_i^{master} < C_i \leq T_i$$
$$C_i^{master} < T_i \Longrightarrow \frac{C_i^{master}}{T_i} < 1$$
$$DBF(\tau_i^{master}, t) = \max(0, \lfloor \frac{t}{T_i} \rfloor C_i^{master})$$
$$DBF(\tau_i^{master}, t) \leq \frac{C_i^{master}}{T_i} t$$
$$DBF(\tau_i^{master}, t) \leq t$$

For the group of constrained deadline segments τ^{cd}, it is the same as in the second case, then we can use the previously calculated DBF (Equation (8)):

$$DBF(\tau_i^{stretched}, t) \leq DBF(\tau_i^{master}, t) + DBF(\{\tau_i^{cd}\}, t)$$
$$DBF(\tau_i^{stretched}, t) \leq t + \frac{(q_i - 1)P_i}{T_i - \eta_i} t$$
$$DBF(\tau_i^{stretched}, t) \leq \frac{C_i}{T_i - \eta_i} t$$

To sum up, the three cases of $\tau^{stretched}$ by SST share the same DBF. ∎

B. Resource Augmentation Bound

Lakshmanan *et al.* have analyzed the resource augmentation bound for their partitioned scheduling algorithm FJ-DMS (Fork-Join–Deadline-Monotonic Scheduling) [2]. The partitioning of the transformed set of tasks is carried out by FBB-FFD scheduling algorithm proposed by Fisher *et al.* [3]. The schedulability test for FBB-FFD is given by:

$$m \geq \frac{\delta_{sum} + u_{sum} - \delta_{max}}{1 - \delta_{max}} \tag{9}$$

For a processor that is v times faster, the following can be applied:

$$\forall 1 \leq i \leq n,\ \eta_i^v \leq \frac{T_i}{v} \Longrightarrow (T_i - \eta_i^v) \geq T_i(1 - \frac{1}{v})$$
$$C_i^v = \frac{C_i}{v}$$
$$u_{sum}^v = \frac{u_{sum}}{v}$$
$$\delta_{max}^v = \frac{\delta_{max}}{v}$$
$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq m$$
$$\delta_{sum}^v = \max_{t>0}\left(\frac{\sum_{i=1}^{n} DBF(\tau_i^{stretched}, t)}{t}\right)$$
$$\delta_{sum}^v \leq \sum_{i=1}^{n} \frac{C_i^v}{T_i - \eta_i^v}$$

In order to simplify the equation, the following can be applied:

$$u_{sum} = \sum_{i=1}^{n} \frac{C_i}{T_i}$$
$$\delta_{sum}^v \leq \sum_{i=1}^{n} \frac{C_i^v}{T_i - \eta_i^v} \leq \sum_{i=1}^{n} \frac{C_i^v}{T_i(1 - \frac{1}{v})} \leq \frac{1}{v-1} u_{sum}$$

By substituting the previous equations in Equation (9), a taskset is schedulable if:

$$m \geq \frac{\frac{m}{v-1} + \frac{m}{v} - \frac{\delta_{max}}{v}}{1 - \frac{\delta_{max}}{v}}$$

This is an increasing function of δ_{max} for $m \geq \frac{v}{2}$.

The density of any parallel thread with constrained deadline in SST is:

$$\forall f_i \geq 0 \Longrightarrow \delta_i^{max} = \frac{1}{1 + \lfloor f_i \rfloor}$$
$$0 \leq \lfloor f_i \rfloor \Longrightarrow 1 \leq 1 + \lfloor f_i \rfloor$$

then $\delta_i^{max} \leq \frac{1}{1+\lfloor f_i \rfloor} \leq 1$.

Using this in Equation (9) and when $m \geq \frac{v}{2}$, the schedulability is ensured if:

$$m \geq \frac{\frac{m}{v-1} + \frac{m}{v} - \frac{1}{v}}{1 - \frac{1}{v}}$$

Applying the same calculations used on TST in [2], and for all $m \geq \frac{v}{2}$, we got the following result:

$$\Longrightarrow v \geq (2 + \sqrt{2}) \approx 3.42$$

This approximated value of resource augmentation bound for SST is the same as the one in TST, and it means if a taskset is feasible on a m speed processors, then it is guaranteed that the transformation will schedule the same taskset on m processors with 3.42 times faster.

V. Conclusion

In a previous paper [4], we presented SST, which is an algorithm design specially for parallel real-time tasks of FJ model based on another transformation called TST, which can be found in literature. Both algorithms enhance the schedulability of parallel tasks while the segment stretch has a practical implementation advantage gained by preventing the use of segment migration.

All the analysis we provided previously was an extensive simulation as a tool to measure the performance of SST in comparison with the TST, so as to see the effects of the proposed modifications. But in this paper, we provide a detailed analysis to calculate the demand bound function of the SST and to prove that it also has the same approximated resource augmentation bound 3.42, the same as TST. This analysis proved that SST does not only have the same performance as TST, but it also reduced the cost of segment migration.

In the future, we will continue working on parallel tasks of FJ model by providing a response time analysis, taking into account the specific constraints of this model of parallel tasks.

References

[1] "Openmp." [Online]. Available: http://www.openmp.org

[2] K. Lakshmanan, S. Kato, and R. (Raj) Rajkumar, "Scheduling parallel real-time tasks on multi-core processors," in *Proceedings of the 31st IEEE Real-Time Systems Symposium (RTSS)*. San Diego, CA, USA: IEEE Computer Society, November-December 2010, pp. 259–268.

[3] N. W. Fisher, S. K. Baruah, and T. P. Baker, "The partitioned scheduling of sporadic tasks according to static-priorities," in *Proceedings of the 18th Euromicro Conference on Real-time Systems (ECRTS)*. Dresden, Germany: IEEE Computer Society, July 2006, pp. 118–127.

[4] F. Fauberteau, S. Midonnet, and M. Qamhieh, "Partitioned scheduling of parallel real-time tasks on multiprocessor systems," in *Proceedings of the Work in Progress session of the 23rd Euromicro Conference on Real-Time Systems (WiP ECRTS)*, Porto, Portugal, July 2011, p. 4pp.

[5] S. K. Baruah, A. K.-L. Mok, and L. E. Rosier, "Preemptively scheduling hard-real-time sporadic tasks on one processor," in *Proceedings of the 11th IEEE Real-Time Systems Symposium (RTSS)*. Orlando, Florida, USA: IEEE Computer Society, December 1990, pp. 182–190.