Work-already-Published: Linear-time admission control for elastic scheduling

Marion Sudvarg*, Chris Gill†, and Sanjoy Baruah‡
Department of Computer Science & Engineering
Washington University in St. Louis
St. Louis, Missouri
Email: *msudvarg@wustl.edu, †cdgill@wustl.edu, ‡baruah@wustl.edu

***Abstract*—Prior algorithms that have been proposed for the uniprocessor scheduling of systems of elastic real-time tasks have computational complexity quadratic ($O(n^2)$) in the number of tasks n, for both initialization and for admitting new tasks during run-time. In our work-already-published [1], we present a more efficient implementation in which initialization takes quasilinear ($O(n \log n)$), and on-line admission control, linear ($O(n)$), time.**

I. Background

The elastic recurrent real-time workload model [2], [3] provides a framework for dealing with overload by compressing (i.e., reducing) the effective utilizations of individual tasks until the cumulative utilization falls below the utilization bound that can be accommodated. For example, an overloaded multimedia system might reduce its sampling or frame rates.

Each task $\tau_i = (U_i^{\min}, U_i^{\max}, E_i)$ is characterized by the minimum utilization $U_i^{\min}$ that it must be provided and the maximum utilization $U_i^{\max}$ that it is able to achieve, as well as an elasticity parameter E_i that "*specifies the flexibility of the task to vary its utilization*" [2]. Given a system $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$ of n such elastic tasks, the objective is to assign each task τ_i a utilization U_i, $U_i^{\min} \le U_i \le U_i^{\max}$, such that (1) $\sum_{i=1}^{n} U_i$ is as large as possible but bounded from above by a specified constant U_d which denotes the maximum cumulative utilization that can be accommodated; and (2) if $U_i > U_i^{\min}$ and $U_j > U_j^{\min}$ then U_i and U_j must satisfy the relationship[1]

$$\left(\frac{U_i^{\max} - U_i}{E_i}\right) = \left(\frac{U_j^{\max} - U_j}{E_j}\right) \tag{1}$$

A task system Γ for which such U_i exist for all the tasks is said to be *feasible*.

A feasible task system with $E_i > 0$ for all tasks[2] $\tau_i \in \Gamma$ may be partitioned into two classes Γ_{VARIABLE} (those tasks for which $U_i > U_i^{\min}$, and which can therefore have their utilizations "varied" –compressed– further if necessary) and Γ_{FIXED} (those for which $U_i = U_i^{\min}$; i.e., their utilizations are now "fixed"). It has been shown [2, Eqn (8)] that for each $\tau_i \in \Gamma_{\text{VARIABLE}}$

$$U_i = U_i^{\max} - \left(\frac{U_{\text{SUM}} - (U_d - \Delta)}{E_{\text{SUM}}}\right) \times E_i \tag{2}$$

where U_{SUM} and E_{SUM} respectively denote the sum of the $U_i^{\max}$ parameters and the E_i parameters of all the tasks in Γ_{VARIABLE}, and Δ denotes the sum of the $U_i^{\min}$ parameters of all the tasks in Γ_{FIXED}.[3]

Given a set of elastic tasks thus partitioned, the algorithm of [2, Fig. 3] determines feasibility and assigns U_i values. It starts by computing U_i values for the tasks assuming that they are all in Γ_{VARIABLE} — i.e., their U_i values are computed according to Expression 2. If any U_i so computed is observed to be smaller than the corresponding $U_i^{\min}$ then that task is moved from Γ_{VARIABLE} to Γ_{FIXED}, the values of U_{SUM}, E_{SUM}, and Δ are updated to reflect this transfer, and U_i values recomputed for all the tasks. The process terminates if no computed U_i value is observed to be smaller than the corresponding $U_i^{\min}$. It is easily seen that one such iteration (i.e., computing U_i values for all the tasks) takes $O(n)$ time. Since an iteration is followed by another only if some task is moved from Γ_{VARIABLE} to Γ_{FIXED} and there are n tasks, the number of iterations is bounded from above by n. The overall running time for the algorithm is therefore $O(n^2)$.

This same algorithm was also repurposed in [2] for admission control: for determining whether a new task seeking to join an already-executing system could be admitted without compromising feasibility, and if so, recomputing the utilization values for the new task as well as for all preëxisting ones.

Extensions to elastic scheduling that were proposed by Chantem et al. [4], [5] reformulate the problem of determining the utilizations as a quadratic programming problem. This allows the iterative technique in [2] to be applied to a more general class of problems, including systems of constrained-deadline, elastic tasks. However, this reformulation continues to have quadratic time-complexity.

[1]For tasks τ_i having $E_i = 0$, $U_i = U_i^{\min} = U_i^{\max}$, and therefore the relationship needs not be satisfied.

[2]All tasks τ_i with $E_i = 0$ must have $U_i \leftarrow U_i^{\max}$; we assume this is done in a pre-processing step, and the value of U_d updated to reflect the remaining available utilization.

[3]Observe that Δ equals the amount of utilization that is allocated to the tasks in Γ_{FIXED}; therefore $(U_d - \Delta)$ represents the amount available for the tasks in Γ_{VARIABLE}, and $\big(U_{\text{SUM}} - (U_d - \Delta)\big)$ the amount by which the cumulative utilizations of these tasks must be reduced from their desired maximums. As shown in the RHS of Expression 2, under elastic scheduling this reduction is shared amongst the tasks in proportion to their elasticity parameters: τ_i's share is (E_i/E_{SUM}).

II. Overview of Work-already-Published

In our work-already-published [1], we define the attribute ϕ_i for an elastic task τ_i as follows:

$$\phi_i \overset{\text{def}}{=} \left(\frac{U_i^{\max} - U_i^{\min}}{E_i} \right) \tag{3}$$

In [1, Theorem 1], we prove a result that allows us to conclude that in the algorithm of [2, Fig. 3], ***tasks may be "moved" from*** Γ_{VARIABLE} ***to*** Γ_{FIXED} ***in order of their*** ϕ_i ***parameters***. Intuitively, ϕ_i is analogous to the amount of force on the spring system of [2] that induces compression on spring i to exactly its length constraint.

Assuming that the tasks are indexed in a linked list such that $\phi_i \leq \phi_{i+1}$ for all $i, 1 \leq i < n$, we can then simply make a *single* pass through all the tasks from τ_1 to τ_n, identifying, and computing U_i values for, all the ones in Γ_{FIXED} before any of the ones in Γ_{VARIABLE}. With appropriate book-keeping (see the pseudo-code already published in [1, Algorithm 1]) this can all be done in a single pass in $O(n)$ time. The cost of sorting the tasks in order to arrange them according to non-increasing ϕ_i parameters is $O(n \log n)$, and hence dominates the overall run-time complexity: determining feasibility and computing the U_i parameters can be done in $O(n \log n) + O(n) = O(n \log n)$ time.[4]

Admission control – determining whether it is safe to add a new task and recomputing all the U_i parameters if so – requires that the new task be inserted at the appropriate location in the already sorted list of preëxisting tasks — this can be achieved in $O(n)$ time. Once this is done, the U_i values can be recomputed in $O(n)$ time by the pseudo-code already published in [1, Algorithm 1]. Similarly, removing a task from the system and recomputing the U_i values also takes $O(n)$ time since sorting is not needed.

In summary, our work-already-published [1] presents a more efficient implementation of the algorithm of [2, Fig. 3] that determines feasibility and computes the U_i values in $O(n \log n)$ time, and performs admission control in $O(n)$ time.

References

[1] M. Sudvarg, C. Gill, and S. Baruah, "Linear-time admission control for elastic scheduling," *Real-Time Systems*, 2021.

[2] G. C. Buttazzo, G. Lipari, and L. Abeni, "Elastic task model for adaptive rate control," in *IEEE Real-Time Systems Symposium*, 1998.

[3] G. C. Buttazzo, G. Lipari, M. Caccamo, and L. Abeni, "Elastic scheduling for flexible workload management," *IEEE Transactions on Computers*, vol. 51, no. 3, pp. 289–302, Mar. 2002. [Online]. Available: http://dx.doi.org/10.1109/12.990127

[4] T. Chantem, X. S. Hu, and M. D. Lemmon, "Generalized elastic scheduling," in *IEEE International Real-Time Systems Symposium*, 2006.

[5] ——, "Generalized elastic scheduling for real-time tasks," *IEEE Transactions on Computers*, vol. 58, no. 4, pp. 480–495, April 2009.

[4]While we considered the problem of implicit-deadline, elastic tasks, our improved algorithm solves the general class of optimization problems described in [4], and could therefore be applied to constrained-deadline task systems as well.