

EEVDF Proportional Share Resource Allocation Revisited

Steve Goddard Jian Tang
Computer Science & Engineering
University of Nebraska - Lincoln
Lincoln, NE 68588-0115
{goddard,jtang}@cse.unl.edu

Abstract

The Earliest Eligible Virtual Deadline First (EEVDF) proportional share resource allocation algorithm proposed by Stoica et al. provides the theoretical foundations needed to achieve real-time performance in time-shared operating systems. Two problems encountered when trying to apply EEVDF to a time-sharing system with real-time and non-real-time constraints are solved. First, a polynomial-time algorithm is presented that efficiently solves the problem of assigning weights to real-time clients in a system that supports both real-time and non-real-time clients. Second, a method is presented to re-compute virtual deadlines when clients enter or leave a dynamic, proportional share system that is scheduled using the EEVDF algorithm.

1. Introduction

Proportional share resource allocation is used to ensure fairness in resource sharing. A *weight* is associated with each client that specifies the relative *share* of a CPU (or any other resource) that the client should receive with respect to other clients. A share represents a fraction of the resource's capacity that is allocated to a client. The actual share of the resource allocated to the client is dependent on the number of clients competing for the resource and their relative weights. If w is the weight associated with a client and W is the sum of all weights associated with clients in the system, then the share (fraction) of the CPU allocated to the client is $f = w/W$. Thus, as competition for the CPU increases, the fraction of the CPU allocated to any one client decreases. This is in contrast to most real-time client models where each client is guaranteed a fixed, minimum share of the CPU. To fix the share of the CPU allocated to a client in proportional share resource allocation, the client's weight relative to the other client weights must vary as clients are created and destroyed [8].

Stoica, Abdel-Wahab, and Jeffay argue, in [8], that weights and shares are duals of each other in proportional share resource allocation. A non-real-time

client is assigned a fixed weight and it receives a share of the processor that is proportional to the value of its weight relative to the total weight of all clients in the system. A real-time client requires a fixed share of the resource to achieve real-time execution. In a dynamic system where clients are allowed to enter when they need the resource and leave when finished, the weights of real-time clients must be continually adjusted to maintain their required fixed share. Thus, non-real-time clients have fixed weights and variable shares while real-time clients have fixed shares and variable weights.

A major difficulty in mixing real-time and non-real-time clients using proportional share, or any other *fair* resource allocation algorithm, is determining the appropriate weights to assign each client such that the real-time clients are guaranteed fixed shares while the non-real-time clients compete for the remaining share of the resource. We call this the *weight-assignment problem*. Stoica, Abdel-Wahab, and Jeffay propose a solution to the weight-assignment problem, but their solution is not practical to implement in a dynamic system with both real-time and non-real-time clients.

In [7], Stoica *et al.* proposed a proportional share resource allocation algorithm, called Earliest Eligible Virtual Deadline First (EEVDF), and the theoretical foundations needed to achieve real-time performance in dynamic, time-sharing operating systems. With the EEVDF algorithm, client requests are scheduled using deadlines based on the client's weight. We recently discovered, however, that the EEVDF algorithm, as presented in [7], may yield incorrect results if deadlines are not re-computed when a client enters or leaves the dynamic system. (The term client was used in [7] to refer to computational entities such as processes. For consistency, the same term is used in this paper.)

This paper makes two contributions. First, a polynomial-time algorithm is presented that efficiently solves the weight-assignment problem. Second, a method is presented to re-compute virtual deadlines when clients enter or leave a dynamic, proportional share system that is scheduled using EEVDF.

The rest of this paper is organized as follows. Section 2 discusses background information and work in proportional share resource allocation directly related to the EEVDF algorithm. Section 3 presents our solution to the weight assignment problem. Section 4 shows how to re-compute virtual deadlines when a client enters or leaves a system scheduled with the EEVDF algorithm. Section 5 summarizes the paper.

2. Background and Related Work

There have been numerous proportional share resource allocation algorithms proposed in the literature for CPU scheduling (e.g., [1, 4, 5, 7, 9, and 10]). The focus of this paper is the EEVDF algorithm proposed by Stoica, *et al.* [7]. With the EEVDF algorithm, client requests are scheduled according to their eligible times and deadlines in the *virtual-time time domain* (as proposed by Zhang [11] and independently by Parekh and Gallager [6]). Based on the weights of the clients in the system, virtual time is computed; virtual time may progress faster, slower or at the same rate as real time. According to client weights, release time and execution time, the virtual eligible times and virtual deadlines are computed using equations presented in [7] (and restated at the end of this section). The requests are then scheduled by observing the Earliest Eligible Virtual Deadline First rule to ensure that no real-time request is ever late by more than $(q-1)$ time units (in the real-time time domain), where q is the length of the scheduling quantum.

The proportional share resource allocation model presented by Stoica, *et al.* [7] (which is similar to the model used by Demers *et al.* [2]) consists of a quantum-based scheduler, real-time clients, and non-real-time clients. Each client is associated with a weight w_i , which defines the share (fraction) of the processor allocated to client i . As defined in [7], the instantaneous share $f_i(t)$ of client i at time t is

$$f_i(t) = \frac{w_i}{\left(\sum_{j \in A(t)} w_j\right)} \quad (1)$$

where $A(t)$ is the set of active clients in the system at time t . In an ideal *fluid-flow system* client i should be allocated

$$S_i(t_1, t_2) = \int_{t_1}^{t_2} f_i(t) dt \quad (2)$$

units of the resource (CPU) in any interval $[t_1, t_2]$. A fluid-flow system can only be approximated since the resource must be allocated in discrete quanta. The difference between the amount of service time client i receives in practice and what it should receive in an ideal fluid-flow system is defined as its *lag*. At time t , the lag of client i is

$$lag_i(t) = S_i(t_0^i, t) - s_i(t_0^i, t) \quad (3)$$

where t_0^i is the time at which client i becomes active and $s_i(t_0^i, t)$ is the processor time actually allocated to client i in the interval [7].

In this work, non-real-time clients are assumed to arrive with a weight \bar{w}_i and a cost c_i , which represents the execution time required for each release of the non-real-time client i . (Assuming a cost c_i is provided with the non-real-time client does not limit the usefulness of this approach since, if c_i is unknown, one can always assume a cost of q , the quantum length. If the client requires either less or more service time, the situation is handled as described in [7].) Real-time clients arrive with a minimum required processor share f_i and a cost c_i . Client weights used for scheduling are computed using the method presented in Section 3.

A fluid-flow system is approximated by executing clients with a quantum-based scheduler in a virtual-time time domain. (The concept of client execution in a virtual-time time domain was originally proposed by Zhang in [11] and independently by Parekh and Gallager in [6].) Virtual time is computed based on the weights of the clients in the system. Specifically, the virtual time of real time t is defined as

$$V(t) = \int_0^t \frac{1}{\left(\sum_{i \in A(t)} w_i\right)} dt \quad (4)$$

where $A(t)$ is the set of active clients in the system at any instant t and w_i represents the weight associated with client i [7]. Using Equation (4), virtual time slows down when new clients join the competition, and virtual time speeds up when clients leave the system.

Clients are scheduled in the virtual-time time domain using the quantum-based Earliest Eligible Virtual Deadline First (EEVDF) algorithm proposed by Stoica, *et al.* [7]: *a new quantum is allocated to the client which has the eligible request with the earliest virtual deadline*. The virtual eligible time and virtual deadline of request k for client i is computed using Equations (5), (6), and (7) where t_0^i is the time at which client i becomes active [7].

$$ve_{i,1} = V(t_0^i) \quad (5)$$

$$vd_{i,k} = ve_{i,k} + \frac{c_i}{w_i} \quad (6)$$

$$ve_{i,(k+1)} = vd_{i,k} \quad (7)$$

3. Solving the Weight-Assignment Problem

A non-real-time client is assigned a fixed weight and receives a share of the processor that is *proportional* to the value of its weight relative to the total weight of all clients in the system. In contrast, a real-time client requires a *fixed* share of the resource to achieve real-time execution. In a dynamic system, non-real-time clients have fixed weights and variable shares while real-time clients have fixed shares and variable weights since the weights of real-time clients must be adjusted to maintain their fixed share whenever a client enters or leaves the system. A major difficulty in mixing real-time and non-real-time clients using proportional share, or any other *fair* resource allocation algorithm, is determining the appropriate weights to assign each client such that the real-time clients are guaranteed fixed shares while the non-real-time clients compete for the remaining share of the resource. We call this the *weight-assignment problem*.

Stoica, Abdel-Wahab, and Jeffay argue, in [8], that weights and shares are duals of each other and propose a method of calculating weights for real-time clients that, for n real-time clients, results in n equations with n unknowns.

We have developed two simpler methods for calculating client weights in a dynamic system consisting of n real-time clients and m non-real-time clients [3]. The first method uses the weights of the non-real-time clients and the fraction of the processor allocated to the real-time processes to compute the weights of the real-time clients. The second method uses the given share required by real-time clients as their weight and maps the given weights of non-real-time clients to new weights that preserves the ratio between any two weights. For space considerations, only the second method is presented here.

Let w_i denote the weight of client i , f_i the share of client i , and W the total weight of all clients in the system. Let F denote the total processor utilization of the real-time clients where share f_i is given for each

client i . That is, $F = \sum_{i=1}^n f_i$.

In a proportional share system, the weights of the $m > 0$ non-real-time clients are provided and their share is dependent on the weights of the other clients in the system. Assume that these weights are $w_{n+1}, w_{n+2}, \dots, w_{n+m}$. Observe that the actual value of weight w_i has no impact on resource allocation; it is the relationship between any two weights that is important. Thus, we set $w_i = f_i$ for each real-time client and

$$w_i = \frac{\bar{w}_i}{\sum_{k=n+1}^{n+m} \bar{w}_k} (1 - F) \quad (8)$$

for each non-real-time client where \bar{w}_i denotes the original weight of non-real-time client i . Using our method, if a new client joins the system (or a client leaves the system), the weights of only non-real-time clients must be re-computed using Equation (8) to ensure all non-real-time clients receive proportional shares; the weights assigned to real-time clients remain the same. Observe that Equation (8) uses the original weight \bar{w}_i to proportionally allocate the *unreserved* share $(1-F)$ of the processor to non-real-time clients. This method results in the sum of all weights

$$W = \left(\sum_{i=1}^{n+m} w_i \right) = 1 \quad (\text{as long as there exists at least one}$$

non-real-time client) and $w_i = f_i$ for all clients, real-time and non-real-time. An advantage of having $W = 1$ is that virtual time will always proceed at the same rate as real time.

4. Re-Computing Virtual Deadlines in Dynamic Systems

In this section, new equations (not presented in [7]) are presented that are needed to re-compute deadlines when clients join or leave the system to ensure a proportional share resource allocation when the EEVDF algorithm is used. If the virtual deadlines of clients are not altered, EEVDF will create an *incorrect* schedule in which clients may miss their deadline by more than $(q-1)$ time units [3], which violates Theorem 1 in [7]. (Theorem 1 in [7] states that any request will be late by at most $(q-1)$ time units.)

The EEVDF algorithm, as described in [7], re-computes $lag_i(t)$ using Equations (1), (2) and (3) with the *newly* assigned weights and shares for each client after time t . Only active clients with $ve \geq V(t)$ are eligible for execution, and the EEVDF algorithm selects the client with the earliest eligible virtual deadline to execute for the next quantum. Since the virtual deadlines are not re-computed with the newly assigned client weights, the EEVDF may select the incorrect clients to execute. Thus, the resulting execution may not reflect a proportional share of the resources.

Virtual deadlines can be re-computed at time t (in the real-time time domain) by dividing the remaining service time required to complete the current request of client i by its weight w_i and adding this value to t . That is, if r_i represents the remaining service time required for request k of client i , set $vd_{i,k} = V(t) + r_i/w_i$. The remaining service time, r_i , is computed as

$$r_i = \bar{S}(t_0^i, d_i) - s(t_0^i, t) = S(t_0^i, t) + \bar{S}(t, d_i) - s(t_0^i, t) \\ = lag_i(t) + \bar{S}(t, d_{i,k})$$

where $\bar{S}(t_1, t_2)$ is the service time client i would receive in a fluid-flow system if none of the weights were changed at time t and $d_{i,k}$ is $vd_{i,k}$ in the real-time time domain. Note that in a perfect fluid-flow system, $lag_i(t) = 0$. In the approximation of a fluid-flow system, $lag_i(t)$ represents how far “behind” or “ahead” the client is in its execution from where it should be. The term $\bar{S}(t_1, t_2)$ represents the amount of service time the request “should” have left (and would have left in a perfect fluid-flow system). Thus, we propose re-computing virtual deadlines for every client i when a client enters or leaves the system at time t using Equation (9) after the new client weights have been assigned.

$$vd_{i,k} = V(t) + \frac{lag_i(t) + \bar{S}_i(t, d_{i,k})}{w_i} \quad (9)$$

Theoretically, the virtual deadlines of all clients need to be re-computed when a client joins or leaves the system. However, our weight assignment method is used and at least one non-real-time client is active, the total of all weights remains the same ($W = 1$) and the weights of the real-time clients remain unchanged. Thus, under these circumstances only the virtual deadlines of non-real-time clients need to be re-computed.

5. Summary

The proportional share resource allocation algorithm proposed by Stoica *et al.* in [7] provides the theoretical foundations needed to achieve real-time performance in time-sharing operating systems. In this paper, solutions are provided to problems encountered when trying to apply the EEVDF algorithm to a dynamic, time-sharing system with both real-time and non-real-time constraints. The results reported here are part of a larger (on-going) effort at the University of Nebraska-Lincoln to support mixed real-time and non-real-time clients using a simple rate-based resource allocation specification.

Our work makes two specific contributions. First, a polynomial-time algorithm that solves the weight-assignment problem was presented. Second, equations to re-compute virtual deadlines when clients enter or leave a dynamic, proportional share system were developed. The results presented here are an extension of the results published in [7] and [8]. Moreover, the re-computation of virtual deadlines is required for the theoretical results of [7] to hold in dynamic proportional share systems.

References

- [1] Baruah, S., Gehrke, J., Plaxton, G., *Fast Scheduling of Periodic Clients on Multiple Resources*, Proc. 9th Intl. Parallel Processing Symp., April 1995, pp. 280-288.
- [2] Demers, A., Keshav, S. Shenker, S., *Analysis and simulation of a fair queueing algorithm*, Proceedings of ACM Sigcomm, pp. 1-12, 1989, and Journal of Internetworking Research and Experience, October 1990.
- [3] Goddard, S., Tang, J., Practical Solutions for Proportional Share Resource Allocation Problems, Technical Memorandum, UNL-CSE-2000-514, Department of Computer Science & Engineering, University of Nebraska-Lincoln, October 2000.
- [4] Maheshwari, U., *Charged-based Proportional Scheduling*, Technical Memorandum, MIT/LCS/TM-529, Laboratory for CS, MIT, July 1995.
- [5] Nieh, J., Lam, M.S., *Integrated Processor Scheduling for Multimedia*, Proc. 5th Intl. Workshop on Network and Operating System Support for Digital Audio & Video, Durham, N.H., April 1995, Lecture Notes in Computer Science, T.D.C. Little & R. Gusella, eds., Vol. 1018, Springer-Verlag, Heidelberg.
- [6] Parekh, A. K., Gallager, R. G., *A Generalized Process Sharing Approach to Flow Control in Integrated Services networks-The Single Node Case*, ACM/IEEE Transactions on Networking, Vol. 1, No. 3, pp. 334-357, 1992.
- [7] Stoica, I., Abdel-Wahab, H., Jeffay, K., Baruah, S., Gehrke, J., Plaxton, G., *A Proportional Share Resource Allocation Algorithm for Real-Time, Time-Shared Systems*, Proc. 17th IEEE Real-Time Systems Symp., Dec. 1996.
- [8] Stoica, I., Abdel-Wahab, H., Jeffay, K., *On the Duality between Resource Reservation and Proportional Share Resource Allocation*, Multimedia Computing & Networking '97, SPIE Proceedings Series, Vol. 3020, pp. 207-214, Feb. 1997.
- [9] Waldspurger, C.A., Weihl, W.E., *Lottery Scheduling: Flexible Proportional-Share Resource Management*, Proc. of the First Symp. on Operating System Design and Implementation, pp. 1-12, Nov. 1994.
- [10] Waldspurger, C.A., *Lottery and Stride Scheduling: Flexible Proportional-Share Resource Management*, Ph.D. Thesis, MIT, Laboratory for CS, September 1995.
- [11] Zhang, L., *Virtual Clock: A New Traffic Control Algorithm for Packet-Switched Networks*, ACM Trans. On Comp. Systems, Vol 9, No. 2, pp-101-124, May 1991.