Proportional Share Scheduling in Single-Server and Multiple-Server Computing Systems

D.H.J. Epema and J.F.C.M. de Jongh
Faculty of Information Technology and Systems
Delft University of Technology
P.O. Box 356, 2600 AJ Delft
The Netherlands
epema@cs.tudelft.nl

Abstract

Proportional Share Scheduling (PSS), which is the allocation of prespecified fractions of a certain resource to different classes of customers, has been studied both in the context of the allocation of network bandwidth and of processors. Much of this work has focused on systems with a single scheduler and when all classes of customers are constantly backlogged. We study the objectives and performance of PSS policies for processor scheduling when these conditions do not hold.

1 The Setting

Proportional Share Scheduling (PSS), which is the allocation of prespecified fractions of a certain resource to different classes of customers, has been studied in the contexts of scheduling processors among classes of jobs and of allocating the capacity of a network link to packet streams; in the latter context, PSS is often called *fair queueing*. These two contexts exhibit important differences in characteristics of the environments in which PSS is to be achieved: In the network case, preemption is not allowed, packets have to be sent in the order of arrival, and the time scale is microseconds, while in the processor case, preemption is often possible, jobs may be run simultaneously, and the time scale is at least seconds. The performance metric usually employed for PSS policies is the deviation from some Processor-Sharing (PS) policy which is emulated when each of the customer classes is constantly backlogged. Most of the work on PSS policies has been done for single-server systems. In contrast, we study PSS policies for processors in distributed systems. The two basic questions we study are suitable performance metrics for such policies, especially in the face of insufficient workload of customer classes, inhomogeneous systems, and the absence of job migration, and the tradeoff between the global and local components of such policies.

The motivation for PSS in networks is to deliver Quality-of-Service guarantees to packet streams that carry continuous real-time data such as speech or video. PSS for sharing processors is useful for supporting applications that deliver these data (e.g., (de-)compressing video data), and for fairly sharing large computing facilities.

In our model, each group of jobs has a *required share*, which is the fraction of the total system capacity it is entitled to. There are different reasons why groups may at certain times not receive their required shares, such as a lack of offered workload. Therefore, we define the *feasible group shares* of groups, which are the shares that may reasonably be expected from the system simultaneously. Defining the *obtained shares* of groups in the obvious way, our most important performance metric is the *group share deviation* during a group's busy periods, which measures the relative difference between the feasible and obtained shares, and the maximum of their expected values across all groups. In addition, we consider the capacity lost due to waiting jobs at some processors while other processors are idle, and to the use of slow instead of fast processors.

In our research, we study combinations of different local (uniprocessor) policies such as FCFS, Priority Queueing and various forms of PS, and of static and dynamic global policies. Among the dynamic global policies are Horizontal and Vertical Partitioning (HP and VP), in which each processor tries to deliver the required shares, and processors are dedicated to single groups in proportion to their required shares, respectively, and which may also allow migrations. Among the static ones is random splitting.

For lack of space and because we are reporting on ongoing work, this paper contains only partial results. In particular, we restrict ourselves to queueing-theoretic results (and so we do not deal with HP and VP, for which we have to resort to simulations), and we omit most of the proofs. Review of work on PPS for sharing processors and network bandwidth can be found in [4] and [6], respectively.

2 The Model

In this section we present our model of proportional share scheduling in single-server and multiple-server computing systems.

2.1 Processors and Jobs

We define a general model of distributed systems consisting of uniprocessors of possibly different capacities. In order to model the user behavior of submitting a job and then waiting for the results before resubmitting it with different parameters, we introduce permanent jobs which re-enter the system as soon as they are finished, in addition to the ordinary arrival streams of jobs.

- There are P processors, processor p having capacity c_p, $p = 1, \ldots, P$. We assume that $c_1 \geq c_2 \geq \cdots \geq c_P$. The total capacity of the system is $c = \sum_p c_p$.
- There are G groups of (single-task) jobs. Jobs are either non-permanent or permanent. The non-permanent jobs of group g arrive according to a Poisson arrival process with rate λ_g. A permanent job immediately re-enters the system (with a new service demand) when it finishes.
- The jobs of group g, both the permanent and the non-permanent ones, have service-time distribution $H_g(t)$ with mean s_g on a unit-capacity processor. On processor p, the service-time distribution of group-g jobs is $H_g(c_p t)$. The traffic intensity of group g due to its non-permanent jobs is $\rho_g = \lambda_g s_g / c$. The total traffic intensity is $\rho = \sum_g \rho_g$.
- Group g has a *required share* r_g, which is the fraction of the system capacity group g is entitled to. We assume that $0 < r_g < 1, g = 1, \ldots, G$, and that $\sum_g r_g = 1$.

We denote by $N(N^P), N_g(N_g^P)$ and N_{gp} the total number of (permanent) jobs in the system, the total number of (permanent) jobs of group g in the system, and the total number of jobs of group g on processor p, respectively.

2.2 Local Scheduling Policies

We study five local scheduling policies, which we assume to be the same on every processor. First Come First Served (FCFS) does not need any comment. In Priority Queueing (PQ), the jobs of group g_1 have preemptive priority over the jobs of g_2 if $r_{g_1} > r_{g_2}$, and the jobs of groups with equal required shares share a single FCFS queue at a single priority level. In PQ, it makes no sense to have permanent jobs, unless perhaps at the lowest priority level.

The three other policies are Processor Sharing (PS) and two of its variations, Priority Processor Sharing (PPS) and Group Priority Processor Sharing (GPPS). In PS, all jobs continuously receive service at the same rate. In PPS (sometimes also called Discriminatory or Generalized Processor Sharing), the service rate of a job is proportional to the required share r_g of its group. In GPPS, the total service rate of group g is proportional to r_g, with the jobs of the same group having equal service rates. As a consequence, we can define the *obtained job share* of a job of group g on processor p in a distributed system with total capacity c as

$$\begin{aligned} \text{PS}: &\quad (c \textstyle\sum_h N_{hp})^{-1} c_p \\ \text{PPS}: &\quad (c \textstyle\sum_h N_{hp} r_h)^{-1} r_g c_p \\ \text{GPPS}: &\quad (c \textstyle\sum_{h, N_{hp}>0} r_h)^{-1} n_g^{-1} r_g c_p \end{aligned}$$

For FCFS and PQ, the obtained share of such a job is either c_p/c or 0, depending on whether the job is in service or not. Note that on a uniprocessor with PS, PPS, or GPPS, the service-time distribution of the permanent jobs is irrelevent.

2.3 Global Scheduling Policies

In this paper we consider as a global scheduling policy only random splitting, which is a static policy. In random splitting, a single global job scheduler makes all scheduling decisions and sends an arriving job of group g to processor p according to a fixed probability for every g and p.

2.4 Performance Metrics

Our main performance objective is to have groups obtain their required shares. However, there are various obstacles for groups to do so.

First of all, groups may not offer enough load to the system. For instance, in a homogeneous system with P processors, there is no way to deliver the required share to group g when $N_g/P < r_g$.

Second, even when in inhomogeneous systems each group might obtain its required share separately, this may not be possible simultaneously. As an example, consider a system with $P = 3, c_1 = 2$, and $c_2 = c_3 = 1$. If $r_1 = r_2 = 0.4$ and groups 1 and 2 have one job each in the system, they can both be given a fraction of 0.8 of processor 1 separately, but the jobs of groups 1 and 2 can never be given more than the sum of the capacities of processors 1 and 2, which amounts to a fraction of 0.75 of the total capacity. Job migration would not help here.

Third, the absence of job migration may make it impossible to deliver the required shares. As an example, consider a homogeneous system with two processors and three groups with each a required share of 0.33.. and one job in the system. Without job migration, we cannot do anything else but put two jobs on one processor and one on the other, with the groups of the former two jobs receiving a share of 0.25 each. With job migration we can have one

job spend one third of its time being serviced on either of the processors.

We conclude that we need a definition of a *feasible group share* as a yard stick of how well a share-scheduling policy performs. We define the *feasible group share* f_g of group g as

$$f_g = \min(r_g, N_g/P).$$

We can guarantee shares of N_g/P for every group simultaneously because ...

Theorem *If .., there exists a schedule for ...*

We define the *obtained share group share* o_g of group g as the sum of the obtained shares of its jobs in the system as defined in Section 2.2, and the *total obtained share* o as the sum of the obtained group shares. Because of the definition of obtained job share, the obtained group share and the total obtained share are normalized to be within the range $[0, 1]$.

The *group-share deviation of group g confined to busy periods* is now defined as

$$\Delta_g^{\mathrm{B,GRP}} = \mathbf{E}\left[\frac{f_g - o_g}{f_g}\middle| N_g > 0\right].$$

This metric averages the deviation of shares over some period of time rather than taking the maximum deviation, which is usually done in bandwidth scheduling [6]. The smaller $\Delta_g^{\mathrm{B,GRP}}$ is, the better the performance, with non-positive values indicating a large enough average obtained share. Our overall metric for share compliance is the *group-share deviation confined to busy periods*

$$\Delta^{\mathrm{B,GRP}} = \min\{\Delta_g^{\mathrm{B,GRP}}\}_g$$

This is a worst-case metric (minimizing over all groups), as opposed by the metric proposed by [1], which considers the vector of share deviations of all groups and uses stochastic majorization for performance comparisons.

Finally, defining the total feasible share by

$$f = \frac{1}{c}\sum_{p=1}^{\min(N,P)} c_p,$$

the *capacity loss* in the system defined as

$$\Delta = \mathbf{E}[f - o]. \tag{1}$$

3 Results for Single-Server Systems

In this section we derive some results for uniprocessors; we restrict our attention to FCFS, PS, and PPS. As these local policies are work conserving, capacity loss cannot occur. During a busy period of group g, of course $f_g = r_g$, and so

$$\Delta_g^{\mathrm{B,GRP}} = \mathbf{E}\left[\frac{f_g - o_g}{f_g}\middle| N_g > 0\right] = 1 - \frac{\mathbf{E}[o_g]}{r_g \mathbf{P}[N_g > 0]}. \tag{2}$$

In this section $p(z_1, \ldots, z_G)$ denotes the probability-generating function of the numbers of non-permanent jobs.

Proposition. *In an FCFS single-server system with exponential service-time distributions with the same means for all groups, and in a PS single-server system with exponential service-time distributions with possibly different means for the groups, we have*

$$p(z_1, \ldots, z_G) = \left(\frac{1-\rho}{1 - \sum_g \rho_g z_g}\right)^{N^P+1}. \tag{3}$$

Proof. For FCFS, see [3]. For PS, one can easily show that the coefficients of (3) satisfy the forward Kolmogorov equations.

Proposition. *If in a single-server system with FCFS, PS, or PPS, group g has permanent jobs, its group-share deviation confined to busy periods is given by*

$$\Delta_g^{\mathrm{B,GRP}} = 1 - \frac{\rho_g + s(1-\rho)}{r_g},$$

with s equal to

$$\begin{aligned} \text{FCFS:} \quad s &= N_g^P s_g / \sum_h N_h^P s_h, \\ \text{PS:} \quad s &= N_g^P / N^P \\ \text{PPS:} \quad s &= N_g^P r_g / \sum_h N_h^P r_h \end{aligned}$$

Proof. Use (2) with $\mathbf{E}[o_g] = \rho_g + s(1-\rho)$ and $\mathbf{P}[N_g > 0] = 1$.

Proposition. *If in a single-server system with FCFS, PS, or PPS, group g does not have permanent jobs, its group-share deviation confined to busy periods is given by*

$$\Delta_g^{\mathrm{B,GRP}} = 1 - \frac{\rho_g}{r_g}\left[1 - \left(\frac{1-\rho}{1-(\rho-\rho_g)}\right)^{N^P+1}\right]^{-1}.$$

Proof. By (3), $\mathbf{P}[N_g > 0]$ is equal to

$$1 - p(1, \ldots, 1, 0, 1, \ldots, 1) = 1 - \left(\frac{1-\rho}{1-\rho+\rho_g}\right)^{N^P+1},$$

so, because $E[o_g] = \rho_g$, (2) yields the desired result.

4 Random Splitting in Multiple-Server Systems

We now turn our attention to distributed systems with random splitting as the global scheduling policy. We assume PS as the local policy. Minimization of a weighted sum of the response times of the groups under random splitting has been studied in [2] and [5] with FCFS and with the locally optimal policy dictated by the so-called c/μ-rule on each processor, respectively. In both papers, different processors speeds and different general service-time distributions of groups are allowed.

Our general problem can be posed in the following way. Given the number P of processors and their capacities c_p, the number of groups G and their required shares r_g, their arrival rates λ_g, and their service-time distributions, find the routing matrix π_{gp} of the fractions of jobs of group g sent to processor p such that all queues are stable, and that $\Delta^{\text{GRP,B}}$ or Δ (or both, if that is possible) is minimized. Here we impose the severe restrictions that the system is homogeneous and that there are no permanent jobs, and we only consider the capacity loss. In fact, what we then study is more of a load-balancing problem that a proportional-share scheduling problem.

4.1 Minimizing Capacity Loss

In this section we study the minimization of capacity loss for homogeneous systems without permanent jobs. We will denote the minimum value of Δ by Δ^{MIN}. By (1), and because $\mathbf{E}[o] = \rho$, the minimization of capacity loss amounts to the minimization of the expected value of the feasible share f.

Proposition. *In a homogeneous system without permanent jobs, the capacity loss is minimized when the processor loads are equal.*

Theorem. (a) *In a homogeneous system without permanent jobs, the minimum capacity loss is given by*

$$\Delta^{\text{MIN}} = 1-\rho+\frac{(1-\rho)^P}{P}\sum_{N=0}^{P-1}(P-N)\rho^N\binom{P+N-1}{N}. \quad (4)$$

(b) *For $\rho \le 0.5$ we have*

$$\lim_{P\to\infty}\Delta^{\text{MIN}} = \frac{\rho^2}{1-\rho}. \quad (5)$$

(c) *For $\rho \ge 0.5$ we have*

$$\lim_{P\to\infty}\Delta^{\text{MIN}} = 1-\rho. \quad (6)$$

Proof. (a) The feasible share f is equal to 1, unless $N < P$, in which case $f = (P-N)/P$. Because the probability of having k jobs in an M/M/1 queue is $(1-\rho)\rho^N$, the joint probability of having n_p jobs on processors $p, p = 1, \ldots, P$, with $\sum n_p = N$, is $(1-\rho)^P\rho^N$. Because the binomial coefficient in (4) is the number of ways of putting N jobs on P processors, (4) follows.

(b) The mean number of jobs in an M/M/1 queue is given by $\rho/(1-\rho)$, which is less than or exceeds 1 for $\rho < 0.5$ and $\rho \ge 0.5$, respectively. For increasing values of P, the distribution of the numbers of jobs on the processors will increasingly well reflect the (discrete) distribution of the number of jobs in an M/M/1 queue, and so for $\rho \le 0.5$, $\mathbf{E}[f]$ approaches $\rho/(1-\rho)$. Now by (1) and because $\mathbf{E}[o] = \rho$, (5) follows.

(c) When $\rho \ge 0.5$, $\rho/(1-\rho) \ge 1$, and so $\mathbf{E}[f]$ approaches 1, and by (1), we can conclude (6).

We find the result in (4) more meaningful than the so-called wait-while-idle probability (the probability that at some processor jobs are waiting while other processors are idle) as a measure for the success of load balancing. In Figure .., we show a graph of Δ^{MIN} as a function of ρ for different numbers of processors.

References

[1] O. Abuamsha and N. Pekergin. Comparison of fair queuing algorithms with a stochastic approach. In *Proc. of Mascots '98*, pages 139–144, 1998.

[2] S. Borst. Optimal probabilistic allocation of customer types to servers. In *Proc. of Sigmetrics '95/Performance '95*, pages 116–125, 1995.

[3] O.J. Boxma and J.W. Cohen. The M/G/1 queue with permanent customers. *IEEE J. Selected Areas in Communications*, 9:179–184, 1991.

[4] D.H.J. Epema. Decay-usage scheduling in multiprocessors. *ACM Trans. on Comp. Syst.*, 16:367–415, 1998.

[5] J. Sethuraman and M.S. Squillante. Optimal stochastic scheduling in multiclass parallel queues. In *Proc. of Sigmetrics '99*, pages 93–102, 1999.

[6] H. Zhang. Services disciplines for guaranteed performance service in packet-switching networks. *Proc. of the IEEE*, 83:1374–1396, 1995.