Priority Operators for Fairshare Scheduling

Gonzalo P. Rodrigo, Per-Olov Östberg, and Erik Elmroth

Distributed systems group,
Department of Computing Science, Umeå Univeristy,
SE-901 87, Umeå Sweden
{gonzalo,p-o,elmroth}@cs.umu.se
www.cloudresearch.se

Abstract. Decentralized prioritization is a technique to influence job scheduling order in grid fairshare scheduling without centralized control. The technique uses an algorithm that measures individual user distances between quota allocations and historical resource usage (intended and current system state) to establish a semantic for prioritization. In this work we address design and evaluation of priority operators, mathematical functions to determine such distances. We identify desirable operator characteristics, establish a methodology for operator evaluation, and evaluate a set of proposed operators for the algorithm.

1 Introduction

Fairshare scheduling is a scheduling technique derived from an operating system task scheduler algorithm [1] that prioritizes tasks based on the historical resource usage of the task owner rather than that of the task itself. This technique defines a "fair" model of resource sharing that allows users to receive system capacity proportional to quota allocations irrespective of the number of tasks they have running on the system (i.e. preventing starvation of users with fewer tasks).

In local resource management (cluster scheduler) systems such as SLURM [2] and Maui [3], this prioritization technique is extended to job level and fairshare prioritization is used to influence scheduling order for jobs based on historical consumption of resource capacity. At this level, fairshare is typically treated as one scheduling factor among many and administrators can assign weights to configure the relative importance of fairsharing in systems.

For distributed computing environment such as compute grids [4], a model for decentralized fairshare scheduling based on distribution of hierarchical allocation policies is proposed in [5], and a prototype realization and evaluation of the model is presented in [6]. Based on this work a generalized model for *decentralized prioritization* in distributed computing is discussed in [7], and we here extend on this work and address design and evaluation of *priority operators*: mathematical functions to determine the distance between individual users' quota allocations and historical resource consumption.

2 Decentralized Prioritization

2.1 Model

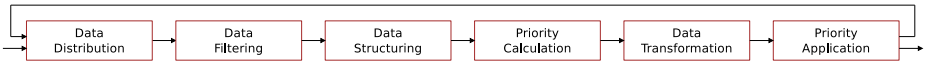


Fig. 1. A computational pipeline for decentralized prioritization. Illustration from [7].

As illustrated in Figure 1, the decentralized prioritization model defines a computational pipeline for calculation and application of prioritization. From a high level, this pipeline can be summarized in three steps: distribution of prioritization information (historical usage records and quota allocations), calculation of prioritization data, and domain-specific application of prioritization (e.g., fairshare prioritization of jobs in cluster scheduling). To model organiza-

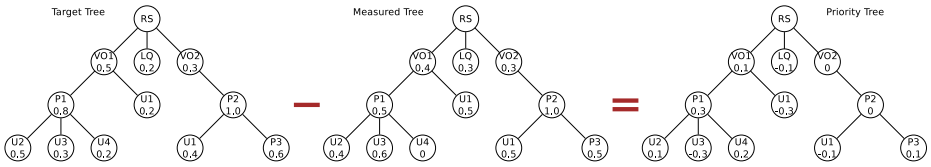


Fig. 2. A tree-based priority calculation algorithm. Tree structure models organization hierarchies (two virtual organizations and a local resource queue). Illustration from [6].

tional hierarchies, the system expresses prioritization target functions (quota allocations or intended system behavior) in tree formats, and prioritization calculation is performed using an algorithm that uses tree structures to efficiently calculate prioritization ordering for users. The tree calculation part of the algorithm is illustrated in Figure 2, where the distance between the intended system state (target tree) and the current system state (measured tree) is calculated via node-wise application of a priority operator (in this case subtraction). The algorithm produces a priority tree - a single data structure containing all priority information needed for prioritization.

For application of prioritization (the last step in the pipeline), priority vectors are extracted from the priority tree and used to infer a prioritization order of the items to be prioritized. In fairshare prioritization of jobs in grid scheduling for example, the target tree contains quota information for users, the (measured) state tree contains a summation of historical usage information for users, and the resulting priority tree contains a measurement of how much of their respective quota each user has consumed. As each user is represented with a unique node in the trees, the values along the tree path to the node can be used to construct a priority vector for the user. Full details of the prioritization pipeline and computation algorithm are available in [6] and [7].

2.2 Challenges / Resolution Limitations

As priority operators lie at the heart of the prioritization algorithm, operator design can have great impact on the semantics of prioritization systems, and further understanding of the characteristics of the priority operator functions is needed, motivating the first part of this work.

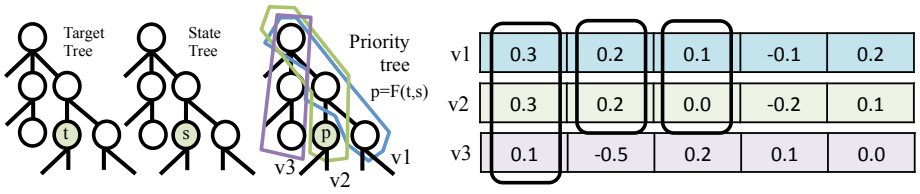
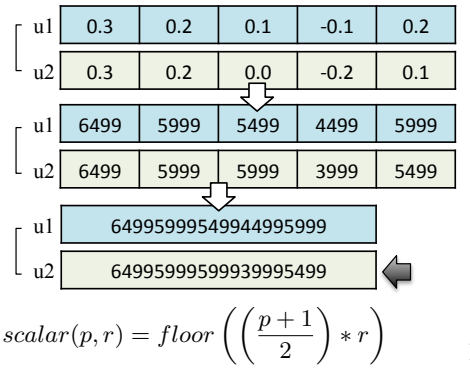


Fig. 3. Construction of priority tree and priority vectors

On the other side, the decentralized fairshare prioritization systems are meant to serve a number of resource management systems in environments created from large complex multilevel organizations. The resources access is governed by policies which structure is mapped from those organization and, as a consequence, these policies have the shape of large trees (both deep and wide). After using the priority operators to create the priority tree (as seen in Figure 2), the tree is traversed from top to bottom extracting the priority vectors (Figure 3) which have independent components representing each "level" of corresponding subgroups in the tree. The vector with the highest value at the most relevant component is the one chosen. For two or more vectors with the same component value, subsequent components are used to determine priority order.



$$scalar(p,r) = floor\left(\left(\frac{p+1}{2}\right) * r\right)$$

Fig. 4. Priority vector serialization, resolution r=10000.

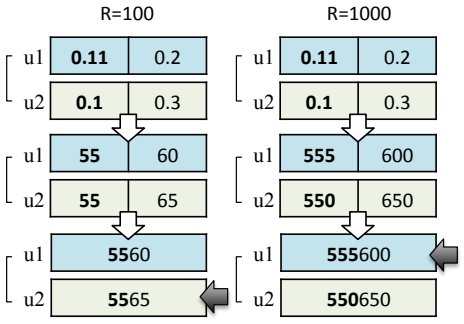


Fig. 5. Resolution limitation impact on the ordering process.

However, the dimension of the overall system is translated to the size of the vector. To simplify the compare operations they are mapped on lexicographic strings. As we can see in Figure 4, for an example scalar resolution of 10000, each component of the vector is linearly mapped to a scalar in the range of 0 to 9999, where -1 is translated to 0 and 1 to 9999. Then, the values are concatenated

to construct the final string. In this example, by comparing the scalar with a simple numeric operation we can determine that u2 has a greater priority than u1 (full process described in [6]). It is important to highlight that the full final string is needed, because any transformation of the vectors that would not keep the individual element presence in it would eliminate the capacity of dividing the share in a true hierarchical way.

However, mapping the priority values to a scalar space with a limited domain has consequences: A number of priority values will map to the same scalar, which may affect the ordering process. In Figure 5 two users have two different priority vectors, when the scalar resolution is 100, the first components of both vectors map to the same scalar, although u1 has a greater priority. The result is that the u2 gets a final bigger scalar string, and thus is selected over u1. When we increase the resolution to 1000 we observe how the ordering becomes correct.

At the same time, it is also important to remember that it is desirable to use the smallest possible resolution. The overall size of the system, will bring thousands of users organized in deep trees, increasing the number of comparisons and elements in each priority vector. Any small reduction in the resolution will have a significant impact on the resources needed to compute the priority vectors.

The behavior of the operators in low resolution has to be understood and modeled, motivating the second part of this work in which we will study its impact on each operator performance and the trade-off between resolution and other characteristics of the system.

3 Operator design

3.1 Operator definition

For fairshare prioritization, we define priority operators as:

1. An operator is a function with two input variables such that:

$$t \in [0,1], s \in [0,1] \Rightarrow F(t,s) \in [-1,1] \tag{1}$$

$$\begin{aligned} F(t,s) = 0 &\iff t = s \\ F(t,s) > 0 &\iff t > s \\ F(t,s) < 0 &\iff t < s \end{aligned} \tag{2}$$

   where t represents a target value, s a (normalized) state value and $t = s$ is the ideal balance axis transecting the operator value space.
2. The function is strictly increasing on target and strictly decreasing on state:

$$\begin{aligned} &\forall t_j, t_i, s_j, s_i, t, s \in (0,1], \\ F(t_j,u) > F(t_i,u) &\iff \quad t_j > t_i \\ F(t_j,s_i) > F(t_j,s_j) &\iff \quad s_j < s_i \\ F(t_j,s) = F(t_i,s) &\iff \quad t_j = t_i \\ F(t,s_i) = F(t,s_j) &\iff \quad s_j = s_i \end{aligned} \tag{3}$$

3. Operator functions are idempotent and deterministic.

3.2 Operator characteristics

Desirable operator characteristics are dependent on application scenarios. In the context of ordering prioritization (ranking) problems it is considered desirable for operators to:

1. Have well-defined boundary behaviors:

$$\begin{aligned} \forall s \in (0,1], t = 0 \implies & \quad F(t,s) = -1 \\ \forall t \in (0,1], s = 0 \implies & \quad F(t,s) = 1 \end{aligned} \tag{4}$$

   so users with target 0 will always get the lowest possible priority (-1) and users with some target but state 0 will get the highest possible priority (1).
2. Subgroup Isolation: Depend only on target and state values of subgroup member nodes.
3. Redistribute unused resource capacity among users in the same subgroup proportionally to their allocations.
4. Be computationally efficient and have minimal memory footprint.
5. Abide by the principle of equivalence: Two operators $F, F'$are equivalent if they would produce the same priority ordering for a set of users knowing their target and state history. Equivalent operators will have the same characteristics for user priority ordering problems. This property is not strictly desirable for an operator but it can be used to assure that all ordering characteristics proved for an operator are automatically proved for the equivalent ones.

3.3 Operators object of study

In this section we will present the operators that are already used in the fairshare prioritization plus one new contribution (Sigmoid) and one brought from the open source cluster scheduler SLURM. The Absolute, Relative, Relative-2 and Combined operators are defined in [6]

1. Absolute: Expressed by the subtraction of the target and the state.

$$d_{Absolute} = t - s \tag{5}$$

2. Relative: Express what proportion of the user's allocation is available.

$$d_{Relative} = \begin{cases} \dfrac{t-s}{t} & s < t \\ 0 & s = t \\ -\dfrac{s-t}{s} & s > t \end{cases} \tag{6}$$

3. Relative exponential: Increases the effects of the Relative.

$$d_{Relative-n} = \begin{cases} \left(\dfrac{t-s}{t}\right)^n & s < t \\ 0 & s = t \\ -\left(\dfrac{s-t}{s}\right)^n & s > t \end{cases} \tag{7}$$

4. Sigmoid: Designed as a vertical inverse of the relative operator.

$$d_{Sigmoid} = \begin{cases} \sin\left(\frac{\pi}{2}\frac{t-s}{t}\right) & s < t \\ 0 & s = t \\ -\sin\left(\frac{\pi}{2}\frac{s-t}{s}\right) & s > t \end{cases} \tag{8}$$

5. Sigmoid Exponential: Increases the effects of the Sigmoid.

$$d_{Sigmoid-n} = \begin{cases} \sqrt[n]{\sin\left(\frac{\pi}{2}\frac{t-s}{t}\right)} & s < t \\ 0 & s = t \\ -\sqrt[n]{\sin\left(\frac{\pi}{2}\frac{s-t}{s}\right)} & s > t \end{cases} \tag{9}$$

6. Combined: Controlled aggregation of the Absolute and Relative operators.

$$d_{Combined} = k \cdot d_{Absolute} + (1-k) d_{Relative-2} \; \forall k \in [0,1] \tag{10}$$

7. SLURM: The operator used by the SLURM scheduling system [8].

$$d_{SLURMOriginal} = 2^{\left(\frac{-s}{t}\right)} \tag{11}$$

Modified to the operator output value range $[-1, 1]$

$$d_{SLURM} = 2^{\left(1+\frac{-s}{t}\right)} - 1 \tag{12}$$

4 Operator evaluation

We investigate each operator for each desirable operator characteristic.

4.1 Operator Definition

First we start with the second point of the definition. For each operator we study the sign of the first derivative when $t \neq s$. For all operators F:

$$\begin{gathered} \forall s, t \in [0,1] \wedge t \neq s : \\ \frac{d(F(t,s))}{d(p)} > 0, \frac{d(F(t,s))}{d(s)} < 0 \end{gathered} \tag{13}$$

assuring the compliance of this part of the definition. Then, as all $F(p,d)$ are strictly increasing on t and strictly decreasing on s, by studying the upper and lower bounds of the input space we can assure the compliance of the output value space:

$$\begin{gathered} F(1,0) \leq 1, F(0,0) = 0, F(0,1) \geq -1 \\ s, t \in [0,1] \wedge t = s \Leftrightarrow F(t,s) = 0 \end{gathered} \tag{14}$$

4.2 Boundary behavior

In Table 1 we observe the priority values in the boundary cases for each operator:

Operator	$t = 0$	$s = 0$
Absolute	$-s$	t
Relative	-1	1
Relative-2.	-1	1
Combined	$-0.5 - \frac{s}{2}$	$0.5 + \frac{t}{2}$
Sigmoid	-1	1
Sigmoid-2	-1	1
SLURM	-1	1

Table 1. Boundary behavior for each operator

The Absolute and Combined operator fail to comply with this property. For the Absolute, the maximum and minimum possible priority are limited in each case by the target value. The Combined operator inherits this behavior from the Absolute component in the operator.

4.3 Subgroup isolation

This property is assured by the definition of the operators: They take into account only the state and target of the corresponding nodes, which are related to the values of the nodes in the same subgroup as the first represents what share of the usage of this subgroup corresponds to this node and the latter what share of the usage should correspond to it. No data out of the subgroup is used to calculate this input values.

4.4 Proportional distribution of unused share

The situation in which a subset of users in a subgroup are not submitting jobs can be understood as an scenario with a new set of target values (virtual target): Eliminating the non submitting users and recalculating the target of the submitting users dividing the non-used share among them in proportion to their original targets. If the system would only operate with the virtual target as the input for the operator, it would converge to that new target. If an operator produces the same ordering with the virtual target (all users submitting jobs) and the old target (but with some users not submitting jobs), then we can state that the operator will bring the system to the virtual target (even if the input is the old target), spreading the unused share proportionally to the user's targets. If we define $\mathbb{T} = \{\text{set of indexes of the users submitting jobs}\}$. The condition to be complied by the operator can be expressed as:

$$\begin{aligned} i, j \in \mathbb{T} : t'_i = \frac{t_i}{\sum_{i \in \mathbb{T}} t_i}, \sum_{i \in \mathbb{T}} t_i \leq 1, t'_i \geq t_i \\ \forall t'_i, t'_j, s_i, s_j \in [0, 1] : \\ 1.\ F(t'_j, s_j) > F(t'_i, s_i) \Rightarrow F(t_j, s_j) > F(t_i, s_i) \\ 2.\ F(t'_j, s_j) = F(t'_i, s_i) \Rightarrow F(t_j, s_j) = F(t_i, s_i) \end{aligned} \tag{15}$$

where t_i the target of user i, s_i the normalized state of user i and t'_i the virtual target of the user i after adding the proportional part of the unused share. Following this reasoning we proved in [9] that the Relative operator complies with this property and that any operator which would produce the same ordering would also comply with this property. As we will see in the following section, the Relative-2, Sigmoid, Sigmoid-2 and SLURM operators are equivalent to the Relative,so we can conclude that they will also distribute proportionally the unused share among the active users of the same subgroup.

4.5 Computationally efficient

Looking into the formulation of the different operators it is obvious that those ones including power, root or trigonometric operations will be more complex in math related cpu cycles. On memory requirements all should perform in a similar way. Still, the real performance of this operators will largely depend on the final implementation, as a consequence we leave this matter for the evaluation of implemented systems.

4.6 Equivalence

We formulated a theorem in [10] that allows to state that two operator are equivalent under one condition: if two operators F, F' Have the same $G(F, t_j, s_j, t_i) = s_i$ so $F(t_j, s_j) = F(t_i, s_i)$, then, they are equivalent and thus, produce the same ordering. We observe that G for the Relative operator is:

$$G(F, t_j, s_j, t_i) = s_j \frac{t_i}{t_j} \tag{16}$$

As we analyze the operators we can state that the Relative exponential, Sigmoid, Sigmoid exponential and SLURM operators share the same G and thus they are equivalent, sharing the same ordering characteristics: Proportional distribution of unused share among peering users. The Absolute and Combined have a different G between them and towards the Relative.

5 Limited output resolution

For the definition of the problem, we consider the number of possible scalar values as the resolution r. Also, under a certain resolution, each priority value p will have a resulting effective priority value $S(p, r)$, understood as the minimum priority value that will have the same corresponding scalar as p. By applying the scalar formula from Figure 4 and composing it with its own inverse function, we can calculate this effective priority value as:

$$S(p, r) = \frac{scalar(p, r)}{r} * 2 - 1 \tag{17}$$

5.1 Methodology

The output resolution problem on each operator will be studied in three steps. In the first place we will present a coarse grained study of how all the possible input pairs (t, s) are divided in sets which elements map on the same priority value. We could argue that for an operator, the bigger the set corresponding to a priory value p, the smaller resolving power (capacity to distinguish between two users) around p and vice versa. We will call this the input density study.

The second approach will be a fine grained extension of the previous. We will observe that priority operators are not defined for the full output range $[-1, 1]$ on all the possible input target values. When the input density is calculated for a priority value, it aggregates all the corresponding state values for each input target value (no matter if there is corresponding output or not), averaging the data of the resulting analysis and hiding the local behavior of the operators. For this second approach we will study each target value, analyzing the sizes of the sets of state values that map on the same output priority value. For a given operator, priority p and target t, the bigger the set of state values corresponding to the that p under t,the smaller resolving power and vice versa. We will call this the input local density study.

Finally, we will bring the input local density study to a semi-real scenario. We will define a grid scenario with a time window, resource dimension and a fictitious average job size. Then, we will determine what is the minimum output resolution required for that job to be significant for a certain user to make sure that its corresponding priority value changes. This study will be based on the previous step, as the input local density of a priority, target values can be understood as the minimum amount of normalized state that has to be added to the history of a user to assure that its corresponding priority value changes. We will call this the jobs size analysis.

In all cases the study will be focused on certain output ranges that are significant to the system: Around balance, where the state value of user is close to its target; under target, when the state is far under the target; and over target, when the state is far over the target.

5.2 Input density analysis

Calculation method For this analysis we needed to calculate the relationship between the input values corresponding to an effective priority value and the complete input range. The method follows a geometry approach when possible. In Figure 6 we can observe the effect of the discretization on the output of the Absolute operator: for each priority value p there is one horizontal surface related to the set of (t, s) that produced that effective priority value. The area of each surface will represent the relative size of that set, as a consequence, what we are looking for: the input density corresponding to p. When the geometrics of the surface were not simple enough, we used sampling to study the number of input values corresponding the the same priority value.

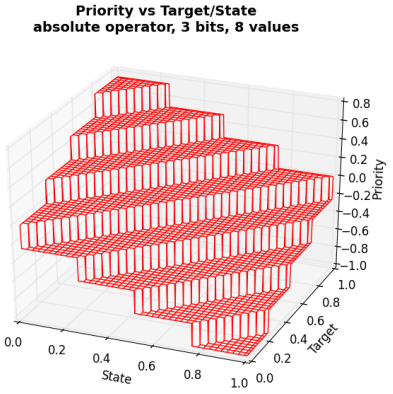


Fig. 6. 3D Representation of effective priorities for the Absolute operator, output resolution 3 bits (8 values).

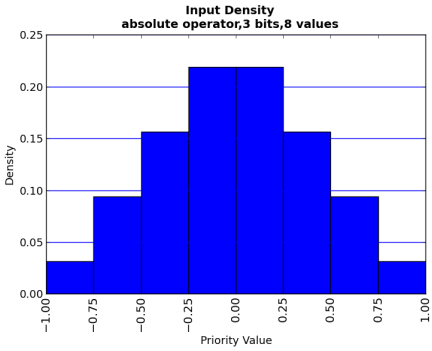


Fig. 7. Input density, Absolute operator, 3 bits, 8 values.

Input density results In order to generate all the input density maps, we run experiments for all the resolutions between 8 and 1048576, in order to ease this description we will talk from now on about the bits needed to represent a given resolution, in this case from 3 to 20 bits. Also, as we increased resolution we noticed something expected, the differences between the priority values became less and less significant. However, for the lower resolutions the operators kept a similar relationship in the same priority value areas.We have chosen 3 bits (8 values) to present the results as it is enough to show the effects of low resolution on each operator.

The results for the operators are presented in the format shown in Figure 7. The data of all operators is presented in the heat map on Figure 8. We can observe many things in this graph, in the first place there is an symmetric behavior around balance for most of the operators (except for the Combined and SLURM). Also we observe that the Relative operator presents the same input density for all its priority values. That the Sigmoid-2 presents the lowest density around balance while the highest around the over/under target cases. The Absolute operator is the one presenting the lowest density in the over/under target cases. This implies that the Sigmoid-2 operator should present the highest resolving power around balance while the lowest in the case of over usage and under usage. In the case of the Relative it presents the same resolving power along the whole output spectrum.

Considerations about this analysis The input density analysis gives a coarse grained picture of the resolution problem, it presents roughly how the whole input is mapped to the output. However, it fails to demonstrate the particularities of the operators. As we will see in the next section, the operators present different input local density distribution for different target values. As the density

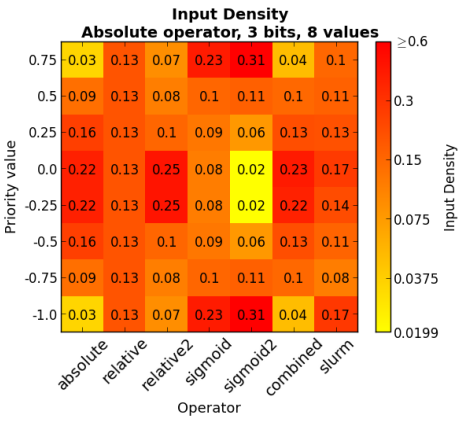


Fig. 8. Absolute Input density comparison among all operators, 3 bits, 8 values.

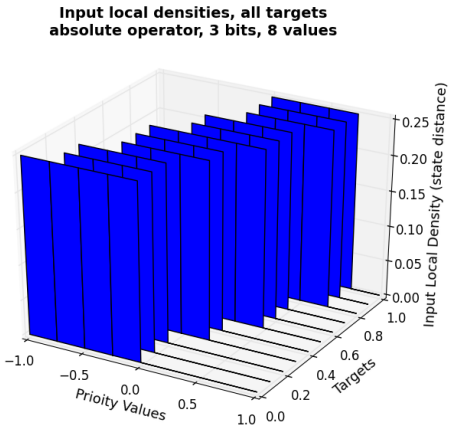


Fig. 9. Aggregation of the input density analysis for different target values, Absolute operator. Resolutions 3 bits.

of a priority value is the normalized aggregation of the local input densities in the whole target ranges, higher values are combined with lower ones, averaging the final result, even more significant as the operators are not covering the full output range for all the target values. Let's illustrate this with an example: the Absolute operator. According to the results in this section it presents a high resolving power around the over/under target areas and low around balance. However, lets look into what happens for each target value (results derived from next section), in Figure 9 we can see the aggregation of input local densities for 11 different targets between 0.0 and 1.0. The first observation is that, as expected, not all targets can generate the full priority range. However, what is more important is that, the resolving power is the same in all cases. This result seems contradictory to the one observed in Figure 7. This apparent divergence comes from the fact that the input density is the aggregation of the input local density along the target range, hiding the local behavior, best cases scenarios and worst case scenarios. We can conclude that the input density view gives an overall picture of the operator behavior but it is incomplete without the per-target input local density study.

5.3 Input local density analysis

Calculation methods In order to calculate the input local density for an operator, we will use an inverse method. For a operator F, a priority p, target t and resolution r, we will compute the lowest and highest state values which produce the effective priority p for target t. The local density will be the difference

between them. This can be expressed as:

$$\begin{aligned} D(F,p,t,r) = |s_j - s_i| : S(p,r) = f \quad & \wedge \\ (\forall s_i \leq s \leq s_j, F(t,s) = f) \quad & \wedge \\ (\forall s < s_i \wedge s_j < s, F(t,s) \neq f) & \end{aligned} \tag{18}$$

In order to calculate those s_j, s_i we will use the inverse expression of the operators on the input s and the set of possible effective priority values for resolution r, $\mathbb{P}_r$. The resulting operation is:

$$D(F,p_i,t,r) = s_{i+1} - s_i = I_s(F,p_{i+1},t) - I_s(F,p_i,t) \tag{19}$$

where:

$$\mathbb{P}_r = \{p_i : i \in \mathbb{N}_r, p_i = -1 + (i-1) \cdot \frac{2}{r}\} \tag{20}$$

(For example $\mathbb{P}_4 = \{-1, -0.5, 0, 0.5\}$) and the inverse function on s of operator F is:

$$I_s(F,p,t) = s : F(t,s) = p \tag{21}$$

Input local density results For this study we run experiments for all the resolutions between 3 and 20 bits. The target range was divided in 10 values, $[0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]$, as the functions are strict increasing on the target, this should be enough to appreciate the general tendency. This conforms 180 operator profiles, which had to be analyzed. One example of the aggregation of all the profiles for the Absolute operator and resolution 3 bits is the Figure 9.

As the resolution was increasing we noticed something expected, the local densities were becoming more similar for a given operator, policy and target. However, for the lower resolutions they kept a similar relationship in the same priority areas. We have chosen 3 bits for the resolution and 0.1, 0.5 as the target values (as they represent one extreme and middle point) to illustrate the behavior of the operators under low resolution.

In Figure 10 we can see one way to represent this data, the operator profile: an overlap between the input density corresponding and the operator plot for the defined target value. It will correlate graphically the priority value, input target, input state and input density, where we consider lower input local densities as something desirable since they indicate higher resolving power. This is the way to interpret them: The horizontal axis represents normalized state from 0.0 to 1.0 while the vertical axis is composed by a range of priority values from -1 to 1. On the graph we represent the overlay of different operators in different colors: The priority values and their corresponding state values in the shape of a plotted line in the corresponding color. On the vertical axis the input density on each priority value in the shape of an horizontal bar, in the corresponding color (its unit is normalized state). In this representation it is possible to observe the different operator function shapes while comparing the different density inputs. In Figure 11 we can observe the contraposition of the Sigmoid-2 and Relative-2,

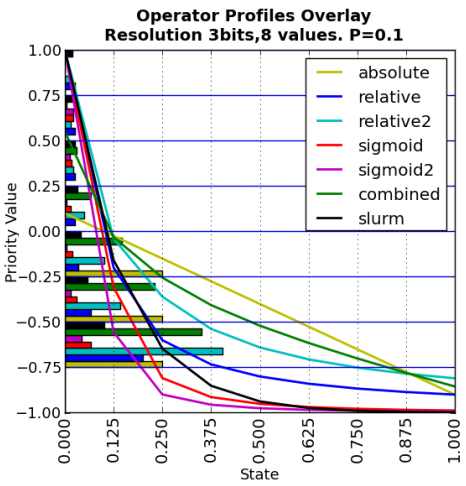


Fig. 10. Operator Profile overlay all operators. Resolution 3 bits. target value 0.1

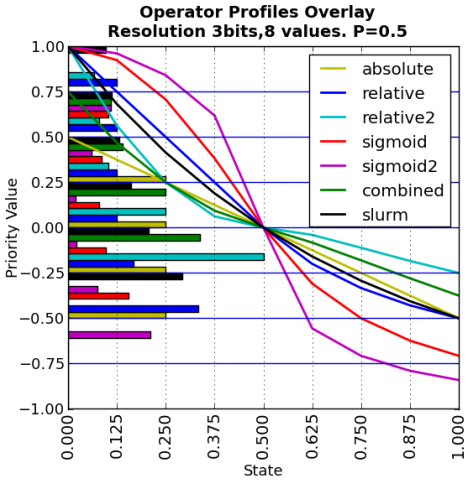


Fig. 11. Operator Profile overlay all operators. Resolution 3 bits. target value 0.5

how the Sigmoid shape is designed to offer the bigger slope (and thus, smaller local density) around balance, and is more flat in the extremes.

As we compare the graphs in Figures 10 and 11 we observe that the range of priority values present in the graphs is smaller as the target increases. This is due to how the operator functions are built: none of them fully covering the range $[-1, 1]$. As the target value increases the most negative value possible for over-state becomes closer to 0.0 (for example, with the Absolute operator, target=0.5 and state = 1.0, the minimum possible priority value is -0.5). This has a first consequence on some of operators: as the target value increases, the same input range of $[0, 1]$ is mapped onto a smaller output range, increasing the overall input local density, increasing the average size of the bars in the graphs. One interesting point is that the Absolute operator presents a constant density in all the graphs and all the priority values, this is due to the subtraction only operation that composes it.

In order to ease the density analysis we mapped the density values on a heat map which opposes the priority values and the operators. The result are Figures 12 and 13. In which the a redder/darker color indicates a higher local density (and lower resolving power) for a pair or operator and target value while a yellower/lighter color implies a lower local density (and higher resolving power). As we observe the graphs, we confirm that, certain operators have a lower input local density behavior for different cases: In balance situations the Sigmoid and Sigmoid-2 presented smaller densities. In under target situations, the Relative-2 operators presented smaller densities. In situations of over target, again the Sigmoid and Sigmoid-2 operators presented lower densities, although not much lower. One intuitive conclusion of this section is that the Sigmoid family presents a higher resolving power for the balance cases and over usage cases, while the Relative operator presents a higher resolving power for the under usage cases.

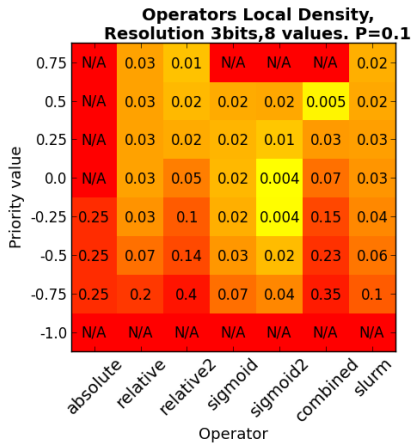


Fig. 12. Input local density heat map. Resolution 3 bits. target value 0.1

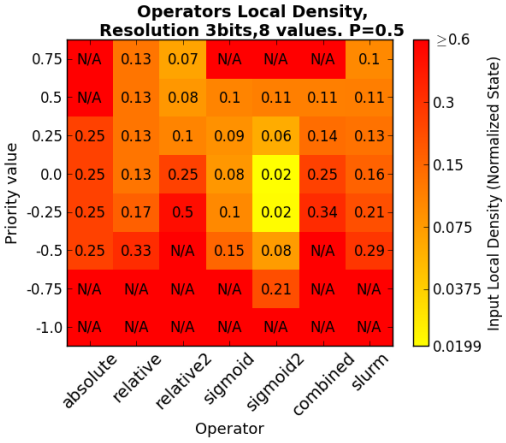


Fig. 13. Input local density heat map. Resolution 3 bits. target value 0.5

5.4 Job Size Analysis

In this study we will bring our analysis to a semi-real environment, a grid scenario with a time window (representing the total historical usage taken into account) and a set of resources. The idea is to understand how big a job has to be to be significant for a user so its effective priority value changes to the next possible one. This will show how many decisions would be made with the same priority values (although new jobs had been completed) or how many bits (minim resolution) would be required for a single job to be significant.

Calculation Method In order to do this analysis we will start from the input local density study. For a given resolution r, operator F, priority p and target t, $D(F, p, t, r)$ can be seen as the minimum amount that the state has to increase to change the output effective priority. This step can me expressed as:

$$\begin{aligned} M(F,t,s,r) = m : m \in (0,1] \quad & \wedge \\ S(F(t,s),r) \neq S(F(t,s+m),r) \quad & \wedge \\ \nexists n : 0 < n < m \wedge S(F(t,s),r) \neq S(F(t,s+n),r) & \end{aligned} \tag{22}$$

This minim step can be translated to a job size, however, it is not trivial: The impact of a job size on the normalized state will depend on the normalized state by itself, as it represents how much of the state pool corresponds to this user: Meaning that one job of a certain size will have a bigger impact for a user with less normalized state than for one with a bigger one. The current state and the state of a user after adding the length of a job can be expressed as:

$$
\begin{aligned}
s_i^d &= \frac{\sum_{0<j\leq d} J_i^j}{U^d}, J_i^{d+1} = k \cdot U^d \\
s_i^{d+1} &= \frac{\sum_{0<j\leq d} J_i^j + J_i^{d+1}}{U^d + J_i^{d+1}} = \frac{\sum_{0<j\leq d} J_i^j + k \cdot U^d}{U^d + k \cdot U^d} \\
&= \frac{(1+k)(\sum_{0<j\leq d} J_i^j + k \cdot U^d)}{U^d} = (1+k)(s_i^d + k)
\end{aligned}
\tag{23}
$$

were s_i^d is the normalized state of a user i at a time d, J_i^j is the size of a job submitted by user i in the time j, U^d is all the state recorded for all users until time d and k is the proportion between the job submitted in time d and the total state recorded until d. This equation is an expression of the new state as a function of the previous state and the proportion between the job and the time window.

Our target is to obtain a function that calculates how big that proportion has to be to jump from one two state values a user (s_i^d) to the next ($s_i^d + 1$). Taking those states as the boundaries for the local density calculation and expressing $s_i^{d+1} - s_i^d$ substituting s_i^{d+1} by the result in the Equation 23 we obtain:

$$
\begin{aligned}
D(F, p_i, t, r) = s_i^{d+1} - s_i^d = (1+k)(s_i^d + k) - s_i^d \\
k^2 + (1 + s_i^d)k - D(F, p_i, t, r) = 0
\end{aligned}
\tag{24}
$$

Which can be solved by using the quadratic equations solving formula:

$$
k = K(F, p_i, t, r, s_i^d) = \frac{-(1+s_i^d) + \sqrt{(1+s_i^d)^2 + 4 \cdot D(F, p_i, t, r)}}{2} \tag{25}
$$

This final result is an expression of k as a function of the operator, previous state, target priority value and resolution. This will be used to calculate what is the minimum job size to be submitted in order to change one user's priority value according to its current state and target and the corresponding operator and resolution. Now that we can find the job size in any case, we established a strategy for the calculations on each operator in each resolution: for each priority, among all the possible k on each target , which is the biggest one (as a bigger k means smaller job). This will allow us to calculate what would be the minimum job size that would assure a change in the effective priority value.

Job size analysis results We took the Swegrid [11] as a reference for this study to create a synthetic example in which the time window is 1 year and the resources managed 600 nodes. This implied that the total state pool of the time window was $U^d = 8760h/n \cdot 600n = 5256000h$. By using that U^d and the method to obtain k we calculated the corresponding job size. We can say that a bigger job size implies a smaller resolving power and a smaller job size implies a bigger resolving power.

For the priority value to study, we focused our calculations in 3 contexts: around balance, where the state is close to the target and the priority values are

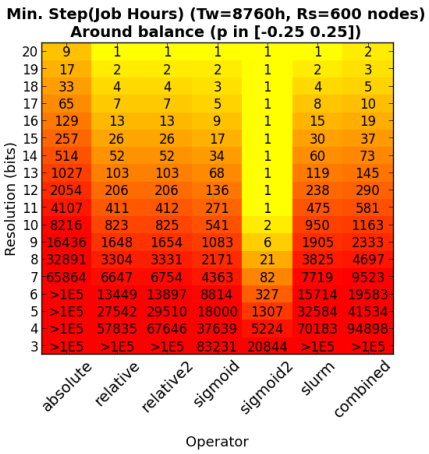


Fig. 14. Job size in hours required to alter user's priority value when state is close to target.

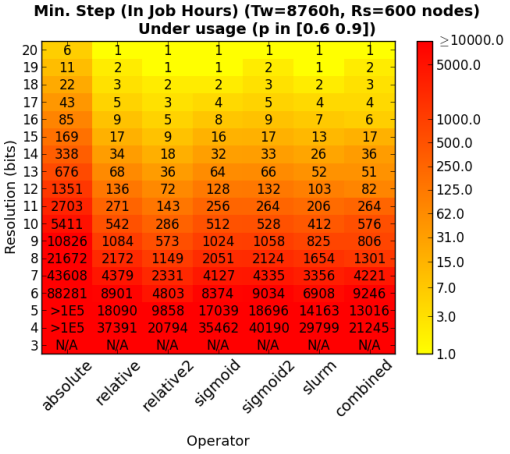


Fig. 15. Job size in hours required to alter user's priority value when state is under target.

in the range $[-0.25, 0.25]$; under target, where priority values are in the range $[0.6, 0.9]$; and over target, where the priorities values are in the range $[-0.9, -0.6]$.

We can observe the results of the around balance study in in Figure 14. This heat map represents the worst case (bigger) among the minimum job size required to make a difference in the priority value for each resolution with each operator. Bigger jobs (and thus smaller resolving power) are represented with darker/redder color while smaller jobs (and thus bigger resolving power) imply a lighter/yellower color. We limited the color range at jobs size 10,000h and used a logarithmic scale for the color distribution. As we can observe the results are independent of the resolution: the Absolute operator needs bigger jobs to make a difference, while the Sigmoid-2 has enough resolving power around balance to always require a smaller job than the rest operators. If we order the operators by job sizes from better to worse resolving power we obtain the following ordering: Sigmoid-2, Sigmoid, SLURM, Relative, Relative-2, Combined, Absolute. This ordering does not change as we increase resolution, although, as it was expected, the job size decreases as the resolution increases.

In the over target scenario presented in Figure 16 we observe the following ordering from better to worse resolving power in all resolutions: Sigmoid-2, Sigmoid, SLURM, Relative, Absolute, Relative-2 and Combined. In the under target scenario presented in Figure 15 we observe the following ordering in all resolutions: Relative-2, Combined, SLURM, Sigmoid, Relative, and Sigmoid-2.

At this point we had a good vision of what was the impact of a certain job for each operator in the studied cases. However we wanted to understand it from a different point of view: For a given job size, what was the minimum resolution in bits needed for an operator to make a difference? We chose an average job size

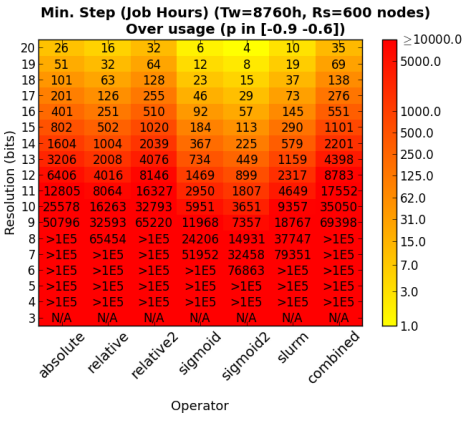


Fig. 16. Job size in hours required to alter user's priority value when state is over target.

Operator	Balance	Over t.	Under t.	Overall
Absolute	12	14	12	14
Relative	9	14	9	14
Relative-2	9	15	8	15
Sigmoid	9	12	9	12
Sigmoid-2	6	11	9	11
SLURM	9	13	8	13
Combined	10	15	8	15

Fig. 17. Output resolution bits required for a job of 2,000h to be significant for a Tw=1 year and Rs=600 nodes. Less is better.

of 2000h (200h,10 nodes). This parameter only affects to the gross value of bits obtained for each operator however, it won't change the relative relationship of the operators.

By parsing the Figures 14, 15 and 16 we produced the results for Figure 17. We can see how the Sigmoid-2 presents a clear advantage around situations of balance. The Sigmoid-2 is the one requiring less bits in the case of balance and over target. For under target, the Relative-2, SLURM and Combined perform better but only with small difference. If we look at the overall picture, the Sigmoid-2 operator is the one which requires less bits in balance and over target while for under target it is just one bit away from the best.

6 Prior and related work

The need of this work was detected in earlier efforts. In [6] and [12], the family of Relative and Combined operator were added as pointed in Section 3. The system's robustness and capabilities were tested and, as a side product, it was discovered that the absolute operator and relative operator generated different ordering when some users did not submit jobs. Also, convergence delays appeared with low resolutions (required for large scale experiments). Those findings are the motivation of this paper.

Fairshare priority is present as a decision factor in well known schedulers. SLURM ([12] shows how our system can substitute SLURM's fairshare engine), as the rest of schedulers, is not meant to deal with as deep hierarchies as Karma ([8], [7]) so the output resolution of its operator is not constrained. SLURM operator, as studied in this paper, complies with all our desired characteristics. Maui [13] presents similar characteristics but uses a version of the Relative as its operator. Other well known example is LSF used by the CERN [14], which choses the absolute operator [15].

The Karma hierarchical system is prepared for much deeper tree schemas than current schedulers and to find work related to the priority resolution we had to look into another scheduling field in which memory is limited: real time schedulers for communication and embedded systems. During the study of the monotonic scheduling algorithm in [16], this problem is named *priority granularity*: "fewer priority levels available than there are task periods", similar to the idea of a number of users with very similar priority and limited resolution around it. In [17], a solution was pointed by creating an exponential grid to distribute the priorities increasing the resolving power around a certain desired area, very similar to what we intended by designing the Sigmoid operator. Finally, [18] presents how the low output resolution can affect negatively to the utilization of resources, as it would alter the best possible decisions, for this case a logarithmic grid is presented. This studies are focused on the final priority value without taking into account the relationship of the input of the prioritization system and its output. In our work we understand how are the system states (input to the priority operator) affected by the limitation on resolution, this allows to modify the system accordingly to the desired resolving power behavior.

7 Future work

This study is a mathematical understanding of the operators in the fairshare prioritization scheduling. it allows to establish boundaries on the mathematical behavior of the functions, however, it would be desired to confront the results with the Karma [7], system: To establish a simulation environment in which all the operators ares tested in a close to real situation, with different output resolutions, tree models (different target values, target tree shapes and depths) and sources of system noise (as presented in [6]). In this context it would interesting to study a possible adaptive operator: a dynamic recommendation of the best suited operator and minimum output resolution depending on the overall state of the system.

We also understand that it would be interesting to test the Relative and Sigmoid operator family on the SLURM scheduling system. Our studies reflect that these operators may have a better resolving power than the SLURM operator, while similar general characteristics. Bringing those conclusion to SLURM, which has a different data pipeline than Karma, would bring light on the compatibility of our operators with other scheduling strategies.

In a different line of thinking, as it was presented in [7], the Karma prioritization engine can be used to deal with scenarios that are not strictly concerned about the ordering of elements, but also about the magnitude of the priority values produced by the system. It would be desirable to understand what is the impact of the different operator output value distribution on this magnitude aware systems.

8 Conclusions

We achieved a deeper understanding on the role of the priority operators in the fairshare prioritization scheduling. We established their core definition and the desired characteristics that emanate from the studied problem. This was followed by the review of the existing operators and the contribution of a new one: The Sigmoid operator.

To evaluate the desired characteristics we established a set methods and mathematical proofs that allow to test the compliance of the existing and future operators. Following this methods, all the presented operators were tested determining which ones complied with the desired characteristics. The results: While the Relative, Relative-n, Sigmoid, Sigmoid-n and SLURM operators comply with all of them, the Absolute and Combined does not on some of them.

At this point our study moved into another dimension, understanding the impact of the limitation in output resolution on the whole prioritization. The first contribution was a three step methodology to evaluate its impact on the resolving power of each operator: the study of the input density, input local density and significant job size. We reviewed the potential of the different possibilities and understood the need to go through the three steps in order to have first, an overall view; second, a local detailed understanding of the operator behavior in all its input domain; and finally, a real world quantification of the relationship between output resolution and resolving power of each operator.

By using this methodology, we were able to compare all the listed operators. We established that the Sigmoid-2 is the one which, in overall, has a better resolving power with less output resolution, being the best in situations of balance and over target. In the case of under target the Relative-2, SLURM and Combined operators are the ones with a better resolving power with low output resolution (although very close to the Sigmoid-2). We would like to highlight that the SLURM operator, taken from the SLURM scheduling system, performs average in the balance and over target situations. One interesting conclusion is that the Sigmoid family, a contribution of this paper, presents the best overall characteristics as a fairshare priority operator.

Finally, this paper establishes a set of research lines to bring these results into the Karma prioritization system.

Acknowledgments

The authors extend their gratitude to Daniel Espling for prior work and technical support, Cristian Klein for feedback and Tomas Forsman for technical assistance. Financial support for the project is provided by the Swedish Government's strategic research effort eSSENCE and the Swedish Research Council (VR) under contract number C0590801 for the project Cloud Control.

References

1. Kay, J., Lauder, P.: A fair share scheduler. Communications of the ACM 31 (1) (1988) 44–55
2. Yoo, A., Jette, M., Grondona, M.: SLURM: Simple Linux Utility for Resource Management. In Feitelson, D., Rudolph, L., Schwiegelshohn, U., eds.: Job Scheduling Strategies for Parallel Processing. Volume 2862 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2003) 44–60
3. Maui Cluster Scheduler: http://www.adaptivecomputing.com/products/open-source/maui/, January 2014.
4. Foster, I., Kesselman, C.: The grid: blueprint for a new computing infrastructure. Morgan Kaufmann (2004)
5. Elmroth, E., Gardfjäll, P.: Design and evaluation of a decentralized system for Grid-wide fairshare scheduling. In H. Stockinger et al, ed.: Proceedings of e-Science 2005, IEEE CS Press (2005) 221–229
6. Östberg, P-O. and Espling, D., Elmroth, E.: Decentralized scalable fairshare scheduling. Future Generation Computer Systems - The International Journal of Grid Computing and eScience 29 (2013) 130–143
7. Östberg, P-O. and Elmroth, E.: Decentralized prioritization-based management systems for distributed computing. In: eScience (eScience), 2013 IEEE 9th International Conference on, IEEE (2013) 228–237
8. Slurm: Multifactor priority plugin - simplified fair-share formula. `https://computing.llnl.gov/linux/slurm/priority_multifactor.html` (January 2014)
9. Rodrigo, G.P.: Proof of compliance for the relative operator on the proportional distribution of unused share in an ordering fairshare system. `http://www8.cs.umu.se/~gonzalo/ShareDemonstration.pdf` (January 2014) (To be published as a technical report).
10. Rodrigo, G.P.: Establishing the equivalence between operators. `http://www8.cs.umu.se/~gonzalo/EquivalenceDemonstration.pdf` (January 2014) (To be published as a technical report).
11. Swegrid: Swegrid organization. `http://snicdocs.nsc.liu.se/wiki/SweGrid` (January 2014)
12. Espling, D., Östberg, P-O. and Elmroth, E.: Integration and evaluation of decentralized fairshare prioritization (aequus) decentralized scalable fairshare scheduling. (2013) (Submitted for publication).
13. Jackson, D., Snell, Q., Clement, M.: Core algorithms of the maui scheduler. In: Job Scheduling Strategies for Parallel Processing, Springer (2001) 87–102
14. CERN: It services - batch service. `http://information-technology.web.cern.ch/services/batch` (January 2014)
15. LSF: Fairshare scheduling. `http://www.ccs.miami.edu/hpc/lsf/7.0.6/admin/fairshare.html` (January 2014)
16. Lehoczky, J., Sha, L., Ding, Y.: The rate monotonic scheduling algorithm: Exact characterization and average case behavior. In: Real Time Systems Symposium, 1989., Proceedings., IEEE (1989) 166–171
17. Sha, L., Lehoczky, J.P., Rajkumar, R.: Task scheduling in distributed real-time systems. In: Robotics and IECON'87 Conferences, International Society for Optics and Photonics (1987) 909–917
18. Lehoczky, J.P., Sha, L.: Performance of real-time bus scheduling algorithms. ACM SIGMETRICS Performance Evaluation Review 14 (1) (1986) 44–53