Supplemental Material for "Analyzing Qualitatively Optimization Objectives in the Design of HPC Resource Manager"

Robin Boëzennec , Fanny Dufossé , Guillaume Pallez

I. Interpretation of standard deviation of the utilization

This section gives an intuition of standard deviation of the utilization and its link with mean utilization based on a simple example.

We consider a platform with N_c resources, and a continuous submission of tasks. Tasks have an average workload equal to W (or work = number of resources used * runtime). We consider two different phases. First from time t=0 to t=T, jobs are submitted with a small arrival rate (=number of jobs submitted by unit of time) λ_1, and then from T to $2T$ jobs are submitted with an higher arrival rate λ_2, with $\lambda_2 < \lambda_1$. The workload arriving in queue per unit of time (called λ(t)) is equal to:

$$\lambda(t) = \begin{cases} \lambda_1 W & \text{if} \quad t \in [0, T] \\ \lambda_2 W & \text{if} \quad t \in [T, 2T] \end{cases}$$

The scheduler performance is expressed by the constant U_m, which is the maximum utilization that it can reach. We suppose that if there are enough submitted jobs, the scheduler always reach this maximum utilization. We also introduce U_i, for i $\in$ {1,2} which corresponds to the minimum utilization required to ensure that the arriving workload is smaller than the machine capacity when the arrival rate is λ_i. As the arriving workload is equal to $\lambda_i W$ and the system capacity to $N_c \cdot U$, we can deduce that $U_i = \frac{W\lambda_i}{N_c}$. As $\lambda_2 < \lambda_1$, we also have $U_2 < U_1$.

Depending on the scheduler maximum utilization of U_m compared to U_1 and U_2, we will have different scheduling configurations. They are shown in Figure 1.

- Case 1: $U_m < \frac{U_1+U_2}{2}$. The schedule does not execute the workload before $2T$, the utilization is constant at U_m and the variance is null.

- Case 2: $U_m \geq U_1$: The schedule executes jobs when they are submitted without putting them in queue. The utilization is equal to U_1 in the first phase and U_2 in the second. This results in a mean utilization of $\frac{U_1+U_2}{2}$ and a utilization variance of $\frac{(U_1-U_2)^2}{4}$.

Manuscript received ... ; revised ... . Robin Boëzennec is with Inria, Labri, University of Bordeaux, and University of Rennes. Guillaume Pallez is with Inria and Université de Rennes. Fanny Dufossé is with Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG.

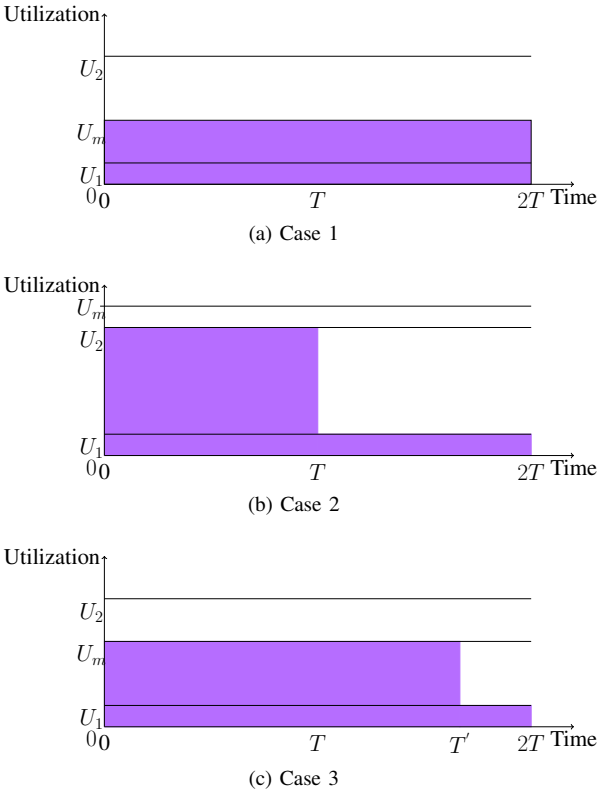


Fig. 1: The different configurations of the scheduling

- Case 3: $\frac{U_1+U_2}{2} \leq U_m < U_1$. The schedule runs all jobs, but some jobs submitted in the first phase are scheduled during the second one. As the scheduler always reach a utilization as high as possible this leads to a utilization of U_m when t $\in$ [0, $T^{'}$] with a certain $T^{'} > T$, and then a utilization of U_2 during [$T^{'}$, $2T$]. In order to compute the mean and variance we need to determine $T^{'}$. At $t = T^{'}$, there are no jobs lefts in the queue, meaning that the workload arrived since $t = 0$ is equal to the quantity of work that has been run by the machine. Hence, we have $\lambda_1 TW + \lambda_2(T^{'} - T)W = U_m T^{'} N_c$, which gives us $T^{'} = T\frac{U_1-U_2}{U_m-U_2}$. We can then compute the mean utilization which is equal to $\frac{U_1+U_2}{2}$ (as we are scheduling a constant workload that fits in a constant time window, it is natural that the mean utilization stays a constant). The utilization variance is equal to $\frac{U_1-U_2}{2}(U_m - \frac{U_1+U_2}{2})$.

Figure 2 summarizes how the mean and variance of uti-

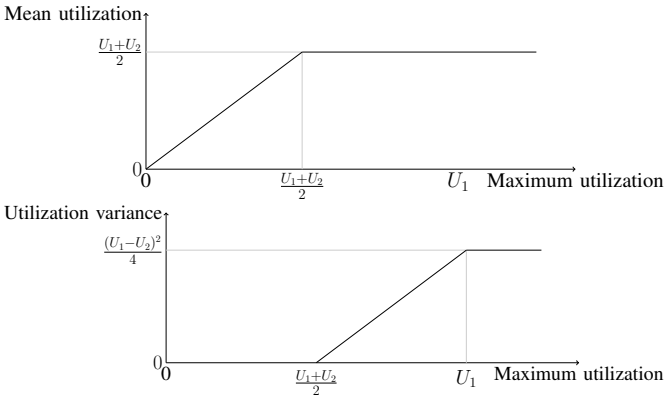


Fig. 2: Mean (left) and variance (right) utilization in function of the maximum utilization U_m.

lization change with the maximum utilization. Once U_m is higher than $\frac{U_1+U_2}{2}$, the mean utilization stops increasing and the utilization variance starts growing.

If we have two scheduling with the same utilization mean, and utilization variances V_1 and V_2, such as $\frac{V_1}{V_2} = \alpha$, and if we make the hypothesis that both scheduling correspond to the second case, we can compute that:

$$U_m^1 = \quad \alpha U_m^2 + (1-\alpha)\frac{U_1+U_2}{2}$$

This allows telling how much an algorithm is better than another. In the same way, if we make the hypothesis that a schedule is in the second case, we can just invert the formula and obtain that $U_m = V\frac{2}{U_1-U_2} + \frac{U_1+U_2}{2}$.

Although these formulas give us numerical values, this is only a particular case, and we are unable to interpret the value of the variance in the general case.

II. Invariance by job permutation of the area-weighted response time

We call the *utilization profile* of a schedule, its function utilization over the time. The aim of this section is to prove the Theorem 1.

Theorem 1. *Given a schedule, if one permutes jobs without changing the utilization profile, the area-weighted mean response (resp. wait) time will keep unchanged.*

To do so, we will define a new metric, and prove some results on this metric. From this first Theorem, we will deduce the desired property on area-Weighted Response Time (WRT). Figure 3 summarizes those results.

This new metric is the mean between the area-weighted response time and the area-weighted wait time. It can therefore be called the time of *half completion* (HC). This means that for a single job, the metric (with its weight) is equal to:

$$HC_i = t_i^{\text{run}}\ N_i^{\text{cores}}\ (t_i^{\text{wait}} + \frac{t_i^{\text{run}}}{2})$$

and the global metric (non-normalized) is equal to:

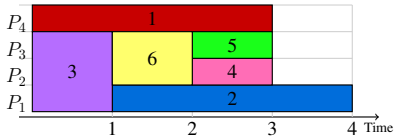
$$\text{HC} = \sum_{i \in J} t_i^{\text{run}}\ N_i^{\text{cores}}\ (t_i^{\text{wait}} + \frac{t_i^{\text{run}}}{2})$$

Where J is the set of all jobs. Note that the half completion time is also weighted by area, but for convenience, we do not precise it in its name.

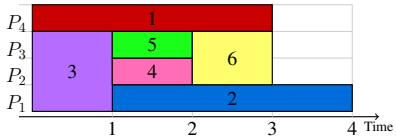
Lemma 1. *: Given a schedule, there are operations that one can apply to the schedule without changing the mean half completion time weighted by work. These operations are:*

- *Permute jobs without changing the utilization profile.*
- *Merge jobs, and set the submission time of the new job as the mean submission time (weighted by work) of the merged jobs.*
- *Split a job in several smaller jobs, such as the mean submission time (weighted by work) of the new job is equal to the submission time of the separated job.*

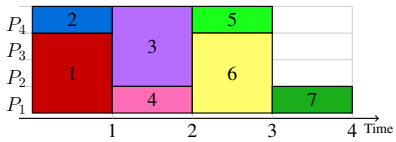
Intuition behind the proof: using half completion time places the point of measure of a job's metric at its barycenter. This allows us to decompose the half completion time of a job as the sum of all the half completion time we obtain by dividing this job in infinitesimally small jobs. We can then swap these infinitesimally small jobs without changing the metric. This allows us to totally change the original job configuration while preserving the half completion time.



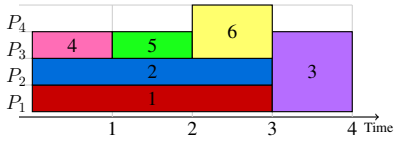
(a) Reference scheduling, WRT = 34, HC = 21.5.



(b) Same jobs, but their order is changed while maintaining the same utilization profile, WRT = 34 (unchanged), HC = 21.5 (unchanged).



(c) Different jobs, but the scheduling has the same utilization profile, WRT = 28 (different), HC = 21.5 (unchanged).



(d) Scheduling made with the same jobs but with a different utilization profile, WRT = 37 (different), HC = 23.5 (different).

Fig. 3: Several scheduling configurations and their corresponding WRT and half completion time (both weighted by work). All jobs are assumed to be queued at t = 0. Area-weighted wait time behaves like the WRT.

Proof: The key of the proof is that HC_i can also be written as the integral of all the wait time that we obtain if we divide the job i in an infinity of infinitesimally small rectangular jobs (all submitted at the same time t_i^{sub}).

$$\begin{aligned}\int_0^{N_i^{\text{cores}}} (\int_{t_i^{\text{wait}}}^{t_i^{\text{wait}}+t_i^{\text{run}}} t dt) dN &= N_i^{\text{cores}} \int_{t_i^{\text{wait}}}^{t_i^{\text{wait}}+t_i^{\text{run}}} t dt \\ &= t_i^{\text{run}}\ N_i^{\text{cores}}\ (t_i^{\text{wait}} + \frac{t_i^{\text{run}}}{2}) \\ &= HC_i\end{aligned}$$

Then we can rewrite HC with this formula:

$$\text{HC} = \sum_{i \in J} \int_0^{N_i^{\text{cores}}} (\int_{t_i^{\text{wait}}}^{t_i^{\text{wait}}+t_i^{\text{run}}} t dt) dN$$

Then, in each of these integrals, we performe the change of variable $t_{\text{old}} = t_{\text{new}} - t_i^{\text{sub}}$. This changes the bound $t_i^{\text{wait}} + t_i^{\text{run}}$ in $t_i^{\text{wait}} + t_i^{\text{run}} + t_i^{\text{sub}} = t_i^{\text{fin}}$, and the bound t_i^{wait} in $t_i^{\text{wait}} + t_i^{\text{sub}} = t_i^{\text{start}}$.

Therefore:

$$\begin{aligned}\text{HC} &= \sum_{i \in J} \int_0^{N_i^{\text{cores}}} (\int_{t_i^{\text{start}}}^{t_i^{\text{fin}}} (t - t_i^{\text{sub}}) dt) dN \\ &= \sum_{i \in J} \int_0^{N_i^{\text{cores}}} \int_{t_i^{\text{start}}}^{t_i^{\text{fin}}} t\ dt dN - \sum_{i \in J} \int_0^{N_i^{\text{cores}}} \int_{t_i^{\text{start}}}^{t_i^{\text{fin}}} t_i^{\text{sub}}\ dt dN\end{aligned}$$

We then define U as surface used by the jobs contained in the set J, and we define C_i as the set of cores on which job i is running. We have:

$$U = \bigcup_{i \in J} (C_i \times [t_i^{\text{start}}, t_i^{\text{fin}}])$$

Where $\times$ represent a Cartesian product. Note that as long as the *utilization profile* doesn't change, the surface U will stay the same. We can then rewrite HC as:

$$\begin{aligned}\text{HC} &= \underbrace{\iint_{\text{U}} t \text{dt dN}}_{A} - \underbrace{\iint_{\text{U}} t_i^{\text{sub}} \text{dt dN}}_{B} \\ &= A - B\end{aligned}$$

A is the integral of a function over the surface U. In addition, this function does not depend on the jobs. So, as long as U is kept unchanged, we can split and merge or permute jobs without affecting A.

B is the integral of another function over the surface U. However this time the function depends on the jobs. In this case, as long as U is kept unchanged, we can reorder jobs without changing the metric. In addition, we can also split and merge jobs following the rules described in the Theorem. This concludes the demonstration of the properties of half completion time. □

We can now deduce the results on WRT (and area-weighted wait time):

Proof: The difference between the half completion time and WRT is equal to:

$$D = \sum_{i \in J} \frac{N_i^{\text{cores}} (t_i^{\text{run}})^2}{2}$$

This is also equal to the difference between wait time (weighted by work) and half completion time. As D depends on the jobs characteristics but not their order, this mean that if we switch jobs order without changing the function u(t), WRT (and wait time weighted by area) will keep unchanged. □