2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum

Online Scheduling for Cloud Computing and Different Service Levels

Uwe Schwiegelshohn
Robotics Research Institute
TU Dortmund University
Dortmund, Germany
uwe.schwiegelshohn@udo.edu

Andrei Tchernykh
Computer Science Department
CICESE Research Center
22830, Ensenada, B.C. México
chernykh@cicese.mx

***Abstract*—In this paper, we address scheduling problems for infrastructure as a service (IaaS). In a typical IaaS scenario, an infrastructure provider offers his resources on demand and with different service levels to his customers. These service levels are mainly distinguished by the amount of computing power a customer is guaranteed to receive within a time frame. In our a model, each service level is described by a slack factor and a price for a processing time unit. If the provider accepts a job it is guaranteed to complete by its deadline, that is its submission time plus its processing time times the slack factor of assigned service level. After a job has been submitted, the provider must decide immediately and irrevocably whether he accepts or rejects the job. We suggest various algorithms and use competitive analysis to discuss different scenarios for this model. These scenarios combine fixed services levels with the single machine model or the parallel identical machines model. Particularly, we demonstrate the benefit of parallelism by showing that we can achieve better competitive factor in a parallel machine scenario than in the corresponding single machine scenario.**

***Keywords*-online scheduling; competitive factors; service levels; cloud computing; slack factor;**

I. Introduction

Cloud Computing has recently become an important computing paradigm. In the Infrastructure-as-a-Service (IaaS) scenario, cloud providers offer computer resources to customers on a pay-per-use base. In order to accommodate the needs of different customers and to generate sufficient income, they may offer different services. Then the price per resource unit depends on the services selected by the customer. In return, the customer receives guarantees regarding the provided resources. These guarantees are also called Quality of Service (QoS) constraints. Often Service level agreements (SLA) are introduced to legally establish the customer-provider relationship. To observe the promised guarantees, QoS constraints must be considered during job scheduling.

On the one hand, the provider wants to offer a large amount of flexibility to his customers. On the other hand, it is hardly possible for a provider and typically inefficient to incorporate all possible QoS constraints into an agreement. Therefore, the provider may select different service categories or service levels. While a large number of service levels leads to high flexibility for the customers, it also produces a significant management overhead. Hence, a suitable tradeoff must be found and adjusted dynamically, if necessary. For instance, systems may offer a small number of specific service levels (e.g. Bronze, Silver, Gold) and allow their customers to select these levels for each job individually to accommodate their needs. Therefore, restrictive QoS guarantees may only be given for a relatively short period of time. To handle this situation, the provider needs algorithms for resource allocation based on these multiple service levels. In this paper, we analyze several simple algorithms for this type of service level scheduling.

The shifting emphasis towards a service-oriented paradigm led to the adoption of SLAs as a very important concept while at the same time, providers and customers were looking for the SLA that is best suited for their needs. There has already been a significant amount of research on various topics related to SLAs. However, this research is almost exclusively executed by practitioners looking for suitable implementations. Little is known about the worst case efficiency of SLA scheduling solutions.

Although there are only very few theoretical results on service level scheduling, results from the real-time scheduling area often show a significant similarity. Therefore, real-time scheduling models can be used as a base for service level scheduling. Nevertheless, these models must be adapted to provide a suitable abstraction of reality on the one hand while on the other hand, key properties of service level computing must be observed to provide benefits for real installations. Since service levels are often considered as a follow up to a service oriented real-time paradigm with deadlines, we start with a simple model that uses a relative job deadline as a function of the job processing time with a constant slack or stretch factor of a service level. Moreover, in order to avoid legal conflicts, the QoS constraints in service level agreements must be easy to monitor. Therefore, information such as latest job completion time (deadline), reserved time for job execution, number of CPUs provided, and price per time unit are well suited to be included in a SLA. Hence, our model represents a valid basic abstraction of a service level in a virtualized infrastructure. Moreover, it can be formalized and treated automatically.

The remainder of the paper is organized as follows: In

978-0-7695-4676-6/12 $26.00 © 2012 IEEE
DOI 10.1109/IPDPSW.2012.130

Section II, we briefly discuss related work with a strong emphasis on hard real-time scheduling. In Section III, we present our notation and formally state our service level scheduling problem. Sections IV and V present known results for a single service level and provide some extensions by considering processing time restrictions. We analyze a scenario with multiple service levels and a single machine in Section VI. Finally, we study the most general scenario of multiple service levels and multiple machines in Section VII.

II. Related Work

Many papers address service levels and quality of service in combination with scheduling problems. Some papers describe and evaluate heuristics, like Yarmolenko and Sakellariou [9]. Other papers present and evaluate software architectures, like Patel et al. [5]. However, we are not aware of a paper that focuses on a theoretical analysis of scheduling with service levels in the context of Cloud computing.

Some theoretical scheduling papers address online scheduling and the rejection of jobs, like Epstein et al. [4]. This paper is interesting in our context as it uses unit-time jobs and therefore incorporates some form of processing time restrictions. However, the objective function is different.

The closest match to our problems can be found in the area of real-time scheduling as we are modeling service levels with the help of deadlines. Baruah and Haritsa [1] discuss the online scheduling of sequential independent jobs on real time systems. In their paper, they present the ROBUST (Resistance to Overload By Using Slack Time) algorithm that guarantees a minimum slack factor for every task. The slack factor f of a task is defined as the ratio of the relative deadline over the execution time requirement. It is a quantitative indicator of the tightness of the task deadline. Therefore, it matches our definition. ROBUST guarantees an effective processor utilization (EPU) of $\frac{f-1}{f}$ during the overload interval which matches our result in Section IV. However, there are differences regarding the objective functions and the proof techniques. Das Gupta and Palis [3] base their hard real-time scenario on the concept of Baruah and Haritsa. Instead of slack factor, they use the expression *stretch factor*. Our paper repeatedly applies the results of Das Gupta and Palis. However, in their scenario, every job has an individual stretch factor with a minimum value while we restrict ourselves to a limited number of fixed slack (stretch) factors. Moreover, there are different prices for computing units in the various service levels while Das Gupta and Palis optimize the total processing time corresponding to just one price. Finally, we slightly extend their proofs by considering restricted processing times.

III. Definitions

In this paper, we consider the single machine model (1) and the parallel identical machine model (P_m). The system

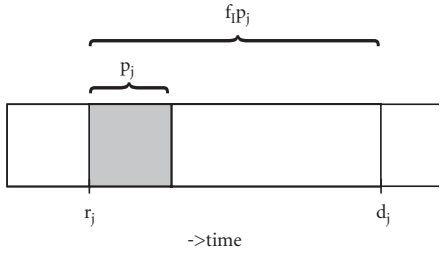

Figure 1. A Job and the Application of the Slack Factor

offers one or more service levels S_i. A service level S_i is associated with a slack factor $f_i > 1$ and the price of a processing time unit u_i. As jobs are submitted over time ($r_{j,\text{online}}$), the processing time p_j and the requested service level S_i of job J_j only become known at its release date $r_j \geq 0$. Furthermore, job J_j has the deadline $d_j = r_j + f_i \cdot p_j$ with f_i being the *slack factor* or *stretch factor* of the service level S_i of the job, see Fig. 1. Intuitively, an accepted job with slack factor f_i will receive on average at least a fraction $\frac{1}{f_i}$ of a machine's resources in the time interval from submission to completion of the job. Immediately at the release date of a job, the infrastructure provider must decide whether this job is accepted for execution or rejected. In order to keep previously given service level guarantees, job J_j can only be accepted if there is a schedule such that neither J_j nor any previously accepted job exceed their deadlines, that is, for all accepted jobs J_i, we have $C_i \leq d_i$ with C_i being the completion time of J_i.

The objective function represents the goal of the infrastructure provider who wants to maximize his total income. Job J_j with service level S_i generates income $v_j = x_j \cdot p_j \cdot u_i$. The binary variable $x_j \in \{0,1\}$ denotes whether job J_j is accepted ($x_j = 1$) or not ($x_j = 0$). Using the three field notation, we can describe our problem either as $1|\text{prmp}, r_{j,\text{online}}, S_i|\sum v_j$ or as $P_m|\text{prmp}, r_{j,\text{online}}, S_i|\sum v_j$ depending on the selected machine model. Note that the problem requires a maximization of the objective function contrary to many other scheduling problems.

We apply competitive analysis, introduced by Tarjan and Sleator [7], to discuss algorithms for these online problems. A maximization method A has a competitive factor $c_V(A) \in [0,1]$ if $c_V(A) = \min_I \frac{V(A)}{V^*} \leq 1$ for all input instances I with $V(A)$ and V^* being the total income generated by method A and the optimal total income, respectively. Due to the maximization of the income, a larger competitive factor is better than a smaller one. Whenever we discuss general upper bounds for competitive factors, we simply us c_V.

After the infrastructure provider has decided on the ac-

ceptance and allocation of a job to a machine, we use the preemptive earliest due date first (preemptive EDD) algorithm to generate a solution for this allocation and for each machine separately. Note that the preemptive EDD algorithm exclusively assigns the machine to a job at any given time while in practice, resources of a machine may be assigned to different jobs in parallel, for instance, if the machine is a multicore processor. Nevertheless, the preemptive EDD algorithm is well suited for our purpose as it is easy to apply and yields an optimal solution for the $1|\text{prmp}, r_{j,\text{online}}|L_{\max}$ problem, see Pinedo [6]. In general, the lateness L_j of job J_j in schedule S is defined to be $\max\{C_j - d_j, 0\}$. Then, we have $L_{\max} = \max_j\{L_j\}$. Remember that for all machine schedules in our problems, $L_{\max} = 0$ must hold as no job can be late. Furthermore, the preemptive EDD algorithm produces a non-delay (greedy) schedule and therefore does not delay the use of resources to the future when yet unknown jobs may need them.

IV. A Single Service Level and a Single Machine

First, we address a simple scenario with a single service level S_I and a single machine, abbreviated SSL-SM. The results of this scenario are useful for more realistic scenarios. This scenario is a special case of hard real-time scheduling.

A. Bounds for SSL-SM

For SSL-SM, Das Gupta and Palis [3] have already presented an upper bound for the competitive factor. In order to consider practical IaaS environments, we introduce a minimal possible processing time $p_{\min} > 0$ and the maximal possible processing time $p_{\max}$ of any submitted job, that is, we assume that the provider sets some limits on the processing time of a job. Furthermore, we allow rational slack factors but restrict the granularity such that either $f_I = \lfloor f_I \rfloor$ or $f_I - \lfloor f_I \rfloor \gg \frac{p_{\min}}{p_{\max}}$ hold. Except for $p_{\min}$ and $p_{\max}$, Theorem 1 is very similar to Theorem 2 of the paper by Das Gupta and Palis and uses the same proof idea. Nevertheless, we repeat the proof as we are using a similar concept in later sections.

Theorem 1: $c_V \leq 1 - (1 - \frac{p_{\min}}{p_{\max}}) \cdot \frac{1}{f_I}$ holds for SSL-SM with service level S_I.

Proof: The adversary submits at time $r_0 = 0$ a job J_0 with processing time $p_0 = p_{\min}$. If J_0 is not accepted then the adversary does not submit any other job resulting in $c_V = 0$.

If job J_0 is accepted and if $f_I \neq \lfloor f_I \rfloor$ holds then the adversary immediately submits another job $J_{\lceil f_I \rceil}$ with

$$\begin{aligned} r_{\lceil f_I \rceil} &= 0 \\ p_{\lceil f_I \rceil} &= (f_I - \lfloor f_I \rfloor) \cdot p_{\max} \\ d_{\lceil f_I \rceil} &= f_I \cdot (f_I - \lfloor f_I \rfloor) \cdot p_{\max} \gg d_0 = f_I \cdot p_{\min}. \end{aligned}$$

If job $J_{\lceil f_I \rceil}$ is not accepted then the adversary does not submit any additional job and we have

$$c_V \leq \frac{p_{\min} \cdot u_I}{((f_I - \lfloor f_I \rfloor) \cdot p_{\max}) \cdot u_I} = \frac{p_{\min}}{(f_I - \lfloor f_I \rfloor) \cdot p_{\max}}.$$

Subsequently, we assume that job $J_{\lceil f_I \rceil}$ is accepted. If $f_I = \lfloor f_I \rfloor$ holds, we simply delete the terms corresponding to job $J_{\lceil f_I \rceil}$. Next, the adversary submits $\lfloor f_I \rfloor$ identical jobs J_j with $1 \leq j \leq \lfloor f_I \rfloor$ and

$$\begin{aligned} r_j &= 0 \\ p_j &= p_{\max} \\ d_j &= f_I \cdot p_{\max} > d_{\lceil f_I \rceil}. \end{aligned}$$

One job J_j with $1 \leq j \leq \lfloor f_I \rfloor$ must be rejected as we have

$$\begin{aligned} p_0 + p_{\lceil f_I \rceil} &+ \lfloor f_I \rfloor \cdot p_{\max} \\ &= p_{\min} + (f_I - \lfloor f_I \rfloor) \cdot p_{\max} + \lfloor f_I \rfloor \cdot p_{\max} \\ &= p_{\min} + f_I \cdot p_{\max} \\ &= d_1 + p_{\min} = \ldots = d_{\lfloor f_I \rfloor} + p_{\min}. \end{aligned}$$

However, in an optimal schedule, job J_0 will be rejected and jobs J_1 to $J_{\lceil f_I \rceil}$ are accepted as they can all complete in time. Therefore, we have

$$\begin{aligned} c_V &\leq \frac{((f_I - 1) \cdot p_{\max} + p_{\min}) \cdot u_I}{f_I \cdot p_{\max} \cdot u_I} \\ &= 1 - (1 - \frac{p_{\min}}{p_{\max}}) \cdot \frac{1}{f_I}. \end{aligned}$$

The combination of the three options yields

$$\begin{aligned} c_V &\leq \max\{0, \frac{p_{\min}}{(f_I - \lfloor f_I \rfloor) \cdot p_{\max}}, 1 - (1 - \frac{p_{\min}}{p_{\max}}) \cdot \frac{1}{f_I}\} \\ &= 1 - (1 - \frac{p_{\min}}{p_{\max}}) \cdot \frac{1}{f_I}. \end{aligned}$$

■

B. Algorithm for SSL-SM

For SSL-SM, Das Gupta and Palis [3] proposed a simple algorithm that we call *greedy acceptance*. This algorithm accepts every new job if this job and all previously accepted jobs can be completed in time. Theorem 2 is similar to the Theorem 3 in the paper of Das Gupta and Palis [3] and shows that the competitive factor of greedy acceptance matches the upper bound of Theorem 1 for unrestricted processing times. However, we only have a single fixed slack factor while Das Gupta and Palis allow arbitrary stretch factors as long as they exceed a minimum value. As our problem is a special case of the hard real-time problem, we can use a simpler proof whose idea will also be used in later sections.

Theorem 2: Greedy acceptance has the competitive factor $1 - \frac{1}{f_I}$ for SSL-SM with service level S_I.

Proof: Let us consider the interval $[0, \max_j d_j)$. This interval is partitioned into a sequence of intervals such that each interval $[a, e)$ consists of a subinterval $[a, b)$ without

any idle time in the schedule (*non-idle subinterval*), and a *completely idle subinterval* $[b, e)$. Any completely idle interval starting at time 0 is ignored as there is no job that can be executed within this interval due to our greedy acceptance policy and the non-delay property of a preemptive EDD schedule.

Consider an arbitrary interval $[a, e)$ as defined above. Assume a job J_0 with $a \leq r_0 < C'_{\max}$. $C'_{\max}$ is the end of the schedule at the time when J_0 is submitted. Clearly, $C'_{\max} \leq b$ holds.

$$\begin{aligned} r_0 + p_0 \cdot f_I &< C'_{\max} + p_0 \Rightarrow \\ p_0 &< \frac{C'_{\max} - r_0}{f_I - 1} \Rightarrow \\ p_0 \cdot f_I &< (C'_{\max} - r_0) \cdot \frac{f_I}{f_I - 1} \end{aligned} \quad (1)$$

is a necessary condition to reject job J_0. Note that Condition 1 is not sufficient as preemption may allow acceptance of J_0 although the condition is not valid. Due to Condition 1, a job J_1 with $a \leq r_1 \leq C'_{\max} \leq b$ and deadline

$$\begin{aligned} d_1 &> r_1 + (C'_{\max} - r_1) \cdot \frac{f_I}{f_I - 1} \\ &= C'_{\max} \cdot \frac{f_I}{f_I - 1} - \frac{r_1}{f_I - 1} \end{aligned}$$

is not rejected by greedy acceptance. Hence, there cannot be any larger non-idle subinterval in interval $[a, e)$ than interval $[a, \max\{e, b \cdot \frac{f_I}{f_I - 1} - \frac{a}{f_I - 1}\})$ for any acceptance policy. As this result holds for all intervals, we have

$$\begin{aligned} c_V(\mathrm{G}_{\text{SSL-SM}}) &\geq \frac{b - a}{b \cdot \frac{f_I}{f_I - 1} - \frac{a}{f_I - 1} - a} \\ &= \frac{f_I - 1}{f_I} = 1 - \frac{1}{f_I}. \end{aligned}$$

■

V. A Single Service Level and Multiple Machines

Next, we extend the infrastructure to many identical machines while there is still only a single service level. Das Gupta and Palis [3] have shown that the upper bound of the single machine model also holds for the parallel identical machine model. Therefore, we only sketch an alternative simple proof for our special case of a single slack factor.

Theorem 3: $c_V \leq 1 - \frac{1}{f_I}$ holds for SSL-PM with service level S_I if the ratio between the largest and the smallest processing time of a job can be arbitrarily large.

Proof: Let us consider only jobs that are submitted at time 0. However, remember that the provider must decide whether a job is accepted before he learns about the next job. If at least one job is allocated to each machine then the adversary submits $m \cdot f_I$ jobs with a sufficiently large processing time p_∞ and deadline $f_I \cdot p_\infty$. Only $f_I - 1$ of these jobs can be executed on each machine resulting in $c_V \leq 1 - \frac{1}{f_I} + \delta$ with $\delta \to 0$ for $p_\infty \to \infty$, see Theorem 1.

Therefore, our algorithm must keep at least one machine idle and guarantee $c_V > 1 - \frac{1}{f_I}$ using only the remaining $m - 1$ machines. We repeatedly apply the same argument to the remaining machines until we end up with a single machine causing a contradiction to Theorem 1. ■

If the processing times of the jobs are restricted then we cannot apply the proof concept of Das Gupta and Palis. However, we can obtain a slightly larger upper bound for SSL-PM with restricted processing times.

Corollary 4: $c_V \leq \frac{f_I}{1 + f_I \cdot (1 - \frac{p_{\min}}{p_{\max}})}$ holds for SSL-PM with service level S_I and its integer slack factor $f_I < \frac{p_{\max}}{p_{\min}}$.

Proof: At time 0, the adversary submits identical (*small*) jobs with processing time $p_{\min}$ until $k \cdot f_I$ jobs have been accepted for some given integer $k < m$ or $m \cdot f_I$ jobs have been submitted. Note that f_I of these small jobs can be executed on each machine. If less than $k \cdot f_I$ of these small jobs are accepted then the adversary does not submit any additional jobs resulting in

$$c_V < \frac{k}{m}. \quad (2)$$

Otherwise the adversary immediately submits $m \cdot f_I$ (*large*) jobs with processing time $p_{\max}$ and release date 0. f_I of the large jobs can be allocated to a machine that does not execute any small job while only $f_I - 1$ of these jobs can be assigned to any other machine. Therefore, at most $m \cdot f_I - k$ large jobs can be accepted while in the optimal schedule all small jobs are rejected and all large jobs are accepted. This results in

$$\begin{aligned} c_V &\leq \frac{(m \cdot f_I - k) \cdot p_{\max} + k \cdot f_I \cdot p_{\min}}{m \cdot f_I \cdot p_{\max}} \\ &= \frac{m \cdot f_I - k}{m \cdot f_I} + \frac{p_{\min}}{p_{\max}} \cdot \frac{k}{m}. \end{aligned} \quad (3)$$

Together, Inequalities 2 and 3 produce

$$c_V \leq \min\{\frac{k}{m}, \frac{m \cdot f_I - k}{m \cdot f_I} + \frac{p_{\min}}{p_{\max}} \cdot \frac{k}{m}\}.$$

For $k = \frac{f_I}{1 + f_I \cdot (1 - \frac{p_{\min}}{p_{\max}})} \cdot m$, the right hand sides of Inequalities 2 and 3 have the same expression and we obtain

$$c_V \leq \frac{f_I}{1 + f_I \cdot (1 - \frac{p_{\min}}{p_{\max}})}.$$

■

A. Algorithm for SSL-PM

Greedy acceptance has the same competitive factor for SSM-SM and SSL-PM, see [3]. It is open whether an algorithm can exploit a possible restriction of processing times. Although the result is known, we provide an extension of the proof of Theorem 1 to the parallel identical machine model as we need this proof later in Section VII

Theorem 5: Greedy acceptance has the competitive factor $1 - \frac{1}{f_I}$ for SSL-PM with service level S_I.

Proof: We use the same approach as in the proof of Theorem 2. Again the schedule interval $[0, \max_j d_j)$ is partitioned into several intervals. Whenever a job starts on an idle machine, a new interval starts. Therefore, within such an interval, it is not possible that an completely idle subinterval is followed by a non-idle subinterval on any machine, that is, for a machine i, an interval $[a, e)$ can either be a non-idle interval ($b_i = e$) or a completely idle interval ($b_i = a$) or a non-idle subinterval $[a, b_i)$ followed by a completely idle subinterval $[b_i, e)$.

Consider an arbitrary interval $[a, e)$ as defined above, and assume that job J_0 is submitted at time r_0 with $a \leq r_0 < b_i$ for all machines i and that job J_0 is rejected. We know from the proof of Theorem 2 that the proof is not affected by distinguishing between the non-idle subinterval at time r_0 and at the final non-idle subinterval. Therefore, we only consider the final values b_i.

$$\begin{aligned} r_0 + p_0 \cdot f_I &< b_i + p_0 \Rightarrow \\ p_0 &< \frac{b_i - r_0}{f_I - 1} \Rightarrow \\ p_0 \cdot f_I &< (b_i - r_0)\frac{f_I}{f_I - 1} \leq (b_i - a)\frac{f_I}{f_I - 1} \end{aligned} \quad (4)$$

must hold for all machines to reject job J_0. Due to Condition 4, a job J_1 with

$$\begin{aligned} d_1 &> r_1 + (b_i - r_1) \cdot \frac{f_I}{f_I - 1} \\ &= b_i \cdot \frac{f_I}{f_I - 1} - \frac{r_1}{f_I - 1} \end{aligned}$$

for at least one machine i is not rejected by greedy acceptance. Due to $a \leq r_1$, there cannot be any larger non-idle subinterval in interval $[a, e)$ than interval $[a, \max\{e, \min_i\{b_i\} \cdot \frac{f_I}{f_I - 1} - \frac{a}{f_I - 1}\})$. As this result holds for all intervals, we have

$$c_V(\mathrm{G}_{\text{SSL-PM}}) \geq 1 - \frac{1}{f_I},$$

see the proof of Theorem 2. ∎

VI. Multiple Service Levels and a Single Machine

In general, IaaS providers offer several service levels. In our model, these levels differ in their slack factors and the price for one processing time unit. Contrary to the hard real-time problem, we use a fixed number of service levels and slack factors. Additionally, our service level include the unit price. Note that our claims only refer to two different service levels S_I and S_{II}. We always require $1 < f_I < f_{II} < \infty$ and $u_I > u_{II}$ for the slack factors and the price values, respectively. Sometimes, additional restrictions are introduced. In most cases, an extension to more than two service levels is possible but will lead to increased complexity without providing additional insights.

In this section, we restrict ourselves again to a single machine. This scenario is abbreviated MSL-SM.

A. Bounds for MSL-SM

For each service level, we can use Theorem 1 to determine an upper bound for the competitive factor. Clearly, the smallest of these upper bounds is an upper bound for the competitive factor for MSL-SM. Furthermore, we can determine an even stronger upper bound for some slack factors. As in Section IV-A, we assume that $f_I - \lfloor f_I \rfloor \gg \frac{p_{\min}}{p_{\max}}$ holds for the slack factor f_I of service level S_I.

Corollary 6:

$$c_V \leq \max\{\frac{\frac{p_{\min}}{p_{\max}}}{(f_I - 1)}, \frac{f_I - 1 + \frac{p_{\min}}{p_{\max}}}{f_I - 1 + \frac{u_I}{u_{II}}}\}$$

holds for MSL-SM with two service levels S_I and S_{II}, and $f_{II} < 2$.

Proof: The proof is similar to the proof of Theorem 1. The adversary submits at time $r_0 = 0$ a job J_0 with processing time $p_0 = p_{\min}$ and service level S_{II}. If J_0 is not accepted then the adversary does not submit any other job resulting in

$$c_V = 0. \quad (5)$$

Otherwise the adversary immediately submits another job J_1 with service level S_{II} and

$$\begin{aligned} r_1 &= 0 \\ p_1 &= (f_I - 1) \cdot p_{\max} \\ d_1 &= f_{II} \cdot (f_I - 1) \cdot p_{\max} \gg d_0 = f_{II} \cdot p_{\min}. \end{aligned}$$

If job J_1 is not accepted then the adversary does not submit any additional job and we have

$$c_V \leq \frac{p_{\min} \cdot u_{II}}{(f_I - 1) \cdot p_{\max} \cdot u_{II}} = \frac{p_{\min}}{(f_I - 1) \cdot p_{\max}}. \quad (6)$$

Otherwise the adversary submits a last job J_2 with service level S_I and

$$\begin{aligned} r_2 &= 0 \\ p_2 &= p_{\max} \\ d_2 &= f_I \cdot p_{\max}. \end{aligned}$$

Note that the relation

$$\begin{aligned} d_2 - d_1 &= (f_I - f_{II} \cdot (f_I - 1)) \cdot p_{\max} > 0 \\ \Leftrightarrow f_{II} &< \frac{f_I}{f_I - 1} \end{aligned}$$

holds. As the function $\frac{x}{x-1} = 1 - \frac{1}{x-1}$ is decreasing for $x > 1$, we have $d_2 > d_1$ due to $1 < f_I < f_{II} < 2$ and $\frac{2}{2-1} = 2$. Therefore, job J_2 must be rejected due to

$$\begin{aligned} p_0 + p_1 + p_2 &= p_{\min} + (f_I - 1) \cdot p_{\max} + p_{\max} \\ &= p_{\min} + f_I \cdot p_{\max} > \max\{d_1, d_2\} = d_2. \end{aligned}$$

However, in an optimal schedule, job J_0 will be rejected and jobs J_1 and J_2 complete in time. Therefore, we have

$$\begin{aligned} c_V &\leq \frac{((f_I-1)\cdot p_{\max}+p_{\min})\cdot u_{II}}{(f_I-1)\cdot p_{\max}\cdot u_{II}+p_{max}\cdot u_I} \\ &= \frac{f_I-1+\frac{p_{\min}}{p_{\max}}}{f_I-1+\frac{u_I}{u_{II}}}. \end{aligned} \quad (7)$$

The combination of the three Inequalities 5, 6, and 7 yields

$$c_V \leq \max\ \{0, \frac{\frac{p_{\min}}{p_{\max}}}{(f_I-1)}, \frac{f_I-1+\frac{p_{\min}}{p_{\max}}}{f_I-1+\frac{u_I}{u_{II}}}\}$$

■

For unrestricted processing times, we have

$$\begin{aligned} c_V &\leq \frac{f_I-1}{f_I-1+\frac{u_I}{u_{II}}} \\ &= 1-\frac{\frac{u_I}{u_{II}}}{f_I-1+\frac{u_I}{u_{II}}} \\ &< 1-\frac{1}{f_I} < 1-\frac{1}{f_{II}} \end{aligned}$$

due to $u_I > u_{II}$ and $1 < f_I < f_{II}$. Therefore, the bound of Corollary 6 is tighter than the bounds of Theorem 1 for the scenario with two service levels and $f_I < f_{II} < 2$. Note that the result of Theorem 1 is obtained for $u_I = u_{II}$. This relation between the Theorem 1 and Corollary 6 suggests that the bound of Corollary 6 also holds for all values of f_I and f_{II}. However, this problem is still open. Note that the determination of such a bound becomes more complex for $f_{II} > 2$. Then the scenario of Corollary 6 comprises at least three jobs with service level S_{II}: Job J_0, job $J_{\lceil f_{II}\rceil}$, and jobs J_1 to $J_{\lfloor f_{II}\rfloor}$. Therefore, the number of options for the determination of an upper bound for the competitive factor increases.

B. Algorithm for MSL-SM

First, we evaluate the performance of the repeatedly used greedy acceptance algorithm for multiple service levels.

Corollary 7: Greedy acceptance has the competitive factor $\frac{u_{II}}{u_I}(1-\frac{1}{f_I})$ for MSL-SM with service levels S_I and S_{II}.

Proof: The proof is similar to the proof of Theorem 2. It uses the same definitions for the intervals $[a,e)$, $[a,b)$, and $[b,e)$. However, subinterval $[a,b)$ may contain jobs with different service levels. Let job J_0 be a job with service level S_I that is submitted at time r_0 with $a \leq r_0 < b$. Note that Condition 1 in the proof of Theorem 2 is also a necessary condition for job rejection in this proof.

In the worst case, there is an optimal schedule such that interval $[a,e)$ contains a non-idle subinterval $[a, b+\frac{b-a}{f_I-1})$, in which only jobs with service level S_I are executed while the algorithm only allocates jobs with service level S_{II} to the interval $[a,b)$. Therefore, we have

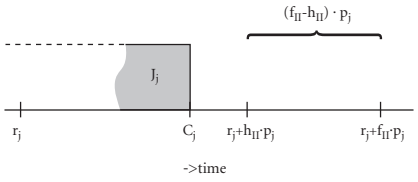

Figure 2. Possible Delay of a Job with Service Level S_{II} due to the Acceptance Factor

$$\begin{aligned} c_V(\mathrm{G_{MSL\text{-}SM}}) &\leq \frac{u_{II}}{u_I(1+\frac{1}{f_I-1})} \\ &= \frac{u_{II}}{u_I}\cdot(1-\frac{1}{f_I}). \end{aligned}$$

■

The competitive factor of Corollary 7 is tight for greedy allocation if the processing times are unrestricted, see Example 8. But contrary to SSL-SM and SSL-PM, a comparison between Corollaries 6 and 7 indicates that the competitive factor of greedy allocation may not be tight in general.

Example 8: Let f_I and f_{II} be integers. At time 0, f_{II} identical jobs with service level S_{II} and processing time $p_1 = \frac{(f_I-1)\cdot p_{\max}+\epsilon}{f_{II}}$ are submitted. Greedy acceptance accepts all jobs as their deadline $d_1 = f_{II}\cdot p$ is identical to their total processing time. Immediately afterwards, f_I identical jobs with service level S_I, processing time $p_2 = p_{\max}$, and release date 0 are submitted. Due to $d_2 = f_I\cdot p_{\max} < d_1+p_2$, we accept any jobs with service level S_I resulting in the ratio

$$\begin{aligned} &\frac{((f_I-1)\cdot p_{\max}+\epsilon)\cdot u_{II}}{f_I\cdot p_{\max}\cdot u_I} \\ = \ &\frac{u_{II}\cdot(f_I-1)}{u_I\cdot f_I}+\frac{u_{II}\cdot\epsilon}{f_I\cdot p_{\max}\cdot u_I}. \end{aligned}$$

Therefore, we consider a different algorithm called *Restricted acceptance* for MSL-SM. This algorithm uses an acceptance factor $h_{II} \geq 1$ and only accepts a job J_j with service level S_{II} only if $d_i - C_i \geq h_{II}\cdot p_i$ holds for all jobs J_i with service level S_{II} and $d_i \geq d_j$. Here, C_i denotes the completion time of job J_i in the preemptive EDD schedule after the inclusion of job J_i.

Intuitively, we replace the slack factor f_{II} by $h_{II} \leq f_{II}$ when deciding whether to accept a new job with service level S_{II}, while the original slack factor is active when a job with service level S_I is submitted. Consider a job J_j with service level S_{II} that completes at time C_j in a schedule that does not execute any job with service level S_I after time r_j. This job can always be delayed for at least time $p_j\cdot(f_{II}-h_{II}) \geq (C_j-r_j)\cdot\frac{f_{II}-h_{II}}{h_{II}}$ by jobs with service level S_I, see Fig. 2.

Corollary 9: Restricted acceptance has the competitive factor $\min\{\frac{h_{II}-1}{f_{II}}, \frac{f_I-1}{f_I} - \frac{h_{II}}{f_{II}} \cdot (1 - \frac{u_{II}}{u_I})\}$ for MSL-SM with service levels S_I and S_{II}, and $h_{II} < f_{II} \cdot (1 - \frac{1}{f_I})$.

Proof: Again the proof uses the same basic concept and the same definitions of intervals $[a, e)$, $[a, b)$, $[a, C'_{\max})$, and $[b, e)$ as in the proof of Theorem 2.

First, we consider an instance in which only jobs with service level S_{II} are submitted. Assume a job J_0 with $a \leq r_0 < C'_{\max}$.

$$\begin{aligned} r_0 + p_0 \cdot h_{II} &< C'_{\max} - r_0 + p_0 \Rightarrow \\ p_0 &< \frac{C'_{\max} - r_0}{h_{II} - 1} \Rightarrow \\ p_0 \cdot f_{II} &< (C'_{\max} - r_0) \cdot \frac{f_{II}}{h_{II} - 1} \end{aligned} \quad (8)$$

is a necessary condition to reject J_0. Due to Condition 8, $a \leq r_0$, and $C'_{\max} \leq b$, no other acceptance policy can generate a larger non-idle subinterval in interval $[a, e)$ than interval $[a, \max\{e, a + (b - a) \cdot \frac{f_{II}}{h_{II}-1}\})$. Therefore, in a schedule that contains only jobs with service level S_{II}, we have

$$c_V(\mathrm{R}_{\text{MSL-SM}}) \geq \frac{h_{II} - 1}{f_{II}}. \quad (9)$$

Now, we consider a job J_1 with service level S_I and $a \leq r_1 \leq b$. Let us further assume that in the present schedule, all jobs in interval $[r_1, b)$ must not complete later than d_1 as job parts that can be executed after d_1 can be ignored with respect to job J_1. Moreover, jobs with service level S_I occupy periods with a total length of g in interval $[a, b)$, see Inequalities 10.

$$(b - r_0) \cdot \frac{f_{II}}{h_{II}} \leq p_0 \cdot f_I < b - r_0 + p_0 + g \Rightarrow \quad (10)$$

$$\begin{aligned} b - r_0 &\leq p_0 \cdot f_I \cdot \frac{h_{II}}{f_{II}} \Rightarrow \\ g &> p_0 \cdot (f_I \cdot (1 - \frac{h_{II}}{f_{II}}) - 1) \end{aligned} \quad (11)$$

is a necessary condition to reject J_1. Note that the use of h_{II} is only beneficial if g in Condition 11 is positive. This yields $h_{II} < f_{II} \cdot (1 - \frac{1}{f_I})$.

Using the same arguments as in the proof of Theorem 2, we have

$$\begin{aligned} c_V(\mathrm{R}_{\text{MSL-SM}}) &\geq \min_g \frac{u_I \cdot g + u_{II} \cdot (p_0 \cdot (f_I - 1) - g)}{u_I \cdot p_0 \cdot f_I} \\ &= \min_g \frac{g \cdot (u_I - u_{II}) + u_{II} \cdot p_0 \cdot (f_I - 1)}{u_I \cdot p_0 \cdot f_I} \\ &\geq 1 - \frac{1}{f_I} - \frac{h_{II}}{f_{II}} \cdot (1 - \frac{u_{II}}{u_I}) \end{aligned} \quad (12)$$

if at least one job with service level S_I is submitted and rejected in the interval. Inequalities 12 and 9 together yield the claim of the corollary. ∎

Note that we cannot easily decide whether Inequality 12 or Inequality 9 is stronger.

Let us compare the results of Corollaries 7 and 9 with the help of an example.

Example 10: Let us assume a scenario with two service levels S_I and S_{II}, and $u_I = 10$, $u_{II} = 1$, $f_I = 2$, $f_{II} = 6$, and $h_{II} = 2 < (1 - \frac{1}{2}) \cdot 6$. Greedy acceptance has the competitive factor $\frac{1}{10} \cdot (1 - \frac{1}{2}) = \frac{1}{20}$ while the competitive factor for restricted acceptance is $\min\{\frac{2-1}{6} = \frac{1}{6}, 1 - \frac{1}{2} - \frac{2}{6} \cdot (1 - \frac{1}{10}) = \frac{1}{5}\} = \frac{1}{6}$.

VII. Multiple Service Levels and Multiple Machines

In this section, we address the scenario MSL-PM with different service levels and parallel identical machines. MSL-PM closely matches real IaaS installations that typically comprise many machines to support scalability. To efficiently exploit this large number of machines, the IaaS provider needs many independent customers. In order to improve flexibility, different service levels are offered.

To analyze algorithms for MSL-PM, we use the results obtained in Sections VI and V. First, we address greedy allocation. As in Section VI, we restrict ourselves to two different service levels although the results can be generalized at the cost of increased complexity.

Corollary 11: Greedy acceptance has the competitive factor $\frac{u_{II}}{u_I}(1 - \frac{1}{f_I})$ for MSL-PM with service levels S_I and S_{II}.

Proof: The proof uses Theorem 5 and Corollary 7. The proof of Theorem 5 extends the interval definitions from a single machine to multiple machines. It shows that greedy allocation cannot benefit from multiple machines as required interval properties must hold for each property separately. For each machine, we apply the proof of Corollary 7. ∎

Moreover, we can extend Example 8 to show that the competitive factor of Corollary 11 is tight for greedy allocation.

However, we can give some service levels a high priority on some machines to obtain a better result. More specifically, jobs with service level S_I can be allocated to every machine while jobs with service level S_{II} can only be allocated to some machines, that is, we use machine eligibility constraints with a so called hierarchical server topology or grade of service, see Bar-Noy et al. [2]. We name this algorithm *Eligible acceptance*. A similar approach has also been used in the context of scheduling parallel jobs in a grid, see Tchernykh et al. [8].

Theorem 12: Eligible acceptance has the competitive factor $\frac{1 - \frac{1}{f_{II}}}{1 + \frac{1 - \frac{1}{f_{II}}}{1 - \frac{1}{f_I}} - \frac{u_{II}}{u_I}}$ for MSL-PM with services levels S_I and S_{II}.

Proof: Jobs with service level S_{II} can only be allocated to k selected machines. Then we obtain

$$c_V(\mathrm{E}_{\text{MSL-PM}}) \geq \frac{k}{m} \cdot (1 - \frac{1}{f_{II}}), \quad (13)$$

due to Theorem 5.

In the worst case, a large number of jobs with service level S_I are submitted just after the jobs with service level S_{II}. The jobs with service level S_I are only allocated to their exclusive set of $m-k$ machines while on the other k machines, their execution is prevented due to the jobs with service level S_{II}. Using Theorem 5 for $m-k$ machines and Corollary 7 for the remaining k machines, we have

$$c_V(\mathrm{E}_{\text{MSL-PM}}) \geq \frac{(k \cdot u_{II} + (m-k) \cdot u_I) \cdot (f_I - 1)}{m \cdot f_I \cdot u_I}. \tag{14}$$

For

$$k = \frac{m}{1 + \frac{1-\frac{1}{f_{II}}}{1-\frac{1}{f_I}} - \frac{u_{II}}{u_I}}$$

and a sufficiently large m such that k is an integer, the right hand sides of Inequalities 13 and 14 yield the same result

$$c_V(\mathrm{E}_{\text{MSL-PM}}) \geq \frac{1-\frac{1}{f_{II}}}{1 + \frac{1-\frac{1}{f_{II}}}{1-\frac{1}{f_I}} - \frac{u_{II}}{u_I}}.$$

■

Again, we can use an extension of Example 8 to show that the competitive factor of Theorem 12 is tight for eligible acceptance.

The competitive factor of eligible acceptance together with a suitable selection of k, see Theorem 12, is always larger than the competitive factor of conventional greedy acceptance, see Corollary 11, as $\frac{u_{II}}{u_I} < 1 < \frac{1-\frac{1}{f_{II}}}{1-\frac{1}{f_I}}$ always holds.

In the following example, we use the values of Example 10.

Example 13: Let us assume a scenario with two service levels S_I and S_{II}, and $u_I = 10$, $u_{II} = 1$, $f_I = 2$, $f_{II} = 6$, and $m = 77$. According to Theorem 12, we select $k = \frac{77}{1+\frac{1-\frac{1}{6}}{1-\frac{1}{2}}+\frac{1}{10}} = 30$. Then eligible acceptance achieves a competitive factor $\frac{30}{77} \cdot (1 - \frac{1}{6}) = \frac{25}{77}$. This competitive factor is better than the competitive factor $\frac{1}{5}$ of restricted acceptance applied to all machines, see Example 10.

VIII. Conclusion

Based on the models in hard real-time scheduling, we have introduced a simple model for service level based job allocation and scheduling. We have discussed scenarios with a single or many identical machines. For these scenarios, we have presented some upper bounds of competitive factors and simple algorithms including their performance evaluation with competitive analysis. It is particularly interesting that a most realistic scenario with two service levels and parallel identical machines has a better competitive factor than the otherwise same scenario using only a single machine. In many other online scheduling problems, the transfer from the single machine model to the parallel identical machine model results in a worse competitive factor. Therefore, this result represents a benefit of parallelism that corresponds to similar claims in practice.

References

[1] S. K. Baruah and J. R. Haritsa, *Scheduling for Overload in Real-Time Systems*, IEEE Transaction on Computers, 46 (9) 1034–1039, 1997.

[2] A. Bar-Noy, A. Freund, and S. Naor, *On-line load balancing in a hierarchical server topology*, SIAM Journal on Computing, 2 (31) 527–549, 2002.

[3] B. Das Gupta and M. A. Palis, *Online Real-time Preemptive Scheduling of Jobs with Deadlines on Multiple Machines*, Journal of Scheduling, Special Issue on Efficient Scheduling Algorithms, 4 (6), 297-312, 2001

[4] L. Epstein, J. Noga, and G. J. Woeginger, *On-line Scheduling of Unit Time Jobs with Rejection: Minimizing the Total Completion Time*, Operations Research Letters, 30 (6), 415–420, 2002.

[5] P. Patel, A. Ranabahu, and A. Sheth, *Service Level Agreement in Cloud Computing*, Cloud Workshops at the International Conference on Object Oriented Programming, Systems, Languages and Applications OOPSLA09, Orlando, Florida, 2009.

[6] M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 2nd ed., Prentice-Hall, New Jersey, 2002.

[7] D. Sleator and R. Tarjan, *Amortized Efficiency of List Update and Paging Rules*, Communications of the ACM 28 (2), 202-208, 1985.

[8] A. Tchernykh, U. Schwiegelshohn, R. Yahyapour, and N. Kuzjurin, *On-line hierarchical job scheduling on grids with admissible allocation*, Journal of Scheduling, 13 (5), 545–552, 2010.

[9] V. Yarmolenko and R. Sakellariou, *An Evaluation of Heuristics for SLA Based Parallel Job Scheduling*, Proceedings of 20th International Parallel and Distributed Processing Symposium IPDPS'06, Rhodes Island, 2006.