Real-time guarantees in flexible advance reservations

Kleopatra Konstanteli, Dimosthenis Kyriazis, Theodora Varvarigou
National Technical University of Athens
Athens, Greece
{kkonst, dkyr, dora}@telecom.ntua.gr

Tommaso Cucinotta, Gaetano Anastasi
Scuola Superiore Sant'Anna
Pisa, Italy
{t.cucinotta, g.anastasi}@sssup.it

***Abstract*—This paper deals with the problem of scheduling workflow applications with Quality of Service (QoS) constraints, comprising real-time and interactivity constraints, over a service-oriented Grid network. A novel approach is proposed, in which high-level Advance Reservations, supporting flexible start and end time, are combined with low-level soft real-time scheduling, allowing for the concurrent deployment of multiple services on the same host while fulfilling their QoS requirements. By undertaking a stochastic approach, in which a-priori knowledge is leveraged about the probability of activation of the application workflows within the reserved time-frame, the proposed methodology allows for the achievement of various trade-offs between the need for respecting QoS constraints (user perspective) and the need for having good resource saturation levels (service provider perspective).**

Keywords - advance reservation; resource management; real-time CPU scheduling.

1. INTRODUCTION

In the last few years, the Service Oriented Architecture (SOA) design methodology has passed from childhood to youth, and has converged with technologies such as Grid Computing [1][2]. The SOA Grid, the resulting converged state of these technologies, holds the promise of addressing some of the most significant problems that are faced in service oriented infrastructures today. However, a long way still divides it from maturity. Focusing on interoperability, reusability and agility, it is often forgotten that provisioning of resources is a critical issue, especially in the case of hundreds of services with real-time constraints spread across various domains. To this direction, the combination of advance reservation and real-time scheduling is considered to be a very promising solution [3][4].

The key objective of the work presented in this paper is to provide a methodology for reliable advance reservation in a SOA Grid environment, with no restrain on the start time that promotes CPU sharing and therefore the overall utilization of the computational resources. Given a new request for advance reservation, the already existing advance reservations and their associated real-time constraints, the proposed algorithm finds all possible co-allocations of the new reservation without breaking the constraints of the reservations made in the past, whose part or even the entire reserved time period may overlap with that of the new one. In doing so, it is necessary to make an estimation of the new reservation's utilization associated with every possible combination of overlapping reservations. Each of these estimations affects the overall mean utilization estimation depending on how probable it is for the corresponding combination of overlapping services to occur. The cost of the offers fluctuates according to this probabilistic guarantee and the provider's business policies. This allows for a probabilistic admission control policy that optimizes the tradeoff between cost and efficiency by not necessarily rejecting an allocation, if the probability of exceeding the timing constraints is acceptable by the user. The methodology applies to pipeline workflows as it is and it is also applicable to non-sequential workflows under the prerequisite that services running in parallel will be scheduled on different CPUs.

The remainder of the paper is structured as follows. A brief overview of related work is given in Section 2. In Section 3, we present the service oriented Grid model our work relies on. This is followed by the development of a probabilistic algorithm for achieving flexible advance reservation in Section 4. In Section 5, we present illustrative numerical examples in order to evaluate the efficiency of the proposed algorithm. Finally, Section 6 concludes the paper.

2. RELATED WORK

The existing studies on advance reservation with more relaxed start time, either assume that the start time and/or end time are static, i.e., the laxity is limited and use rescheduling at run-time, reallocating existing advance reservations at execution time and/or by changing the priority of the running service to ensure that the execution completes prior to its deadline[5][6]. In many cases the algorithms behind these studies are mainly performance-driven and ignore resource sharing. Studies on their performance have shown that the repetition of this action leads in high fragmentation of the scheduling time by increasing the number of time slices that are left unused [7][8]. These time slices can be filled at run-time using techniques such as backfilling [14].

Among the algorithms for real-time scheduling, Rate Monotonic and Earliest Deadline First (EDF) [16] are probably the most widely used techniques in the domain of real-time systems. However, for a proper use in the field of general-purpose processing, they need to be properly enriched by encapsulation techniques, such as *aperiodic servers* [17]. The Proportional Share [9] and Pfair [10] techniques aim to approximate the Generalized Processor Sharing theoretical concept of a *fluid* allocation, in which each application using the resource marks a progress proportional to a given weight. Another approach is based on the concept of Resource Reservations [11], in which the resource allocation for each

The research leading to these results has received funding from the EC 7th Framework Programme FP7 2007/2011 under grant agreement n° 214777, in the context of the IRMOS project (http://www.irmosproject.eu).

application is specified not only in terms of a share, but also of the desired time granularity. The discussion below focuses on the use of a hard reservation variant of the CBS [12], an EDF-based scheduler which provides a strong theoretical foundation and has been proved to be able to cope with aperiodic arrivals.

3. PROBLEM DESCRIPTION

A. System Model

The Grid in our study consists of a number of autonomous sites in each of which a set of computational hosts is participating and can be approximated by the star topology architecture depicted in Figure 1. The center of this service-oriented architecture is occupied by a resource management service, which controls the sharing of the hosts' resources, and is directly connected with them through a wide area network (WAN). As hosts we consider time-shared machines with various processing speeds and soft real-time scheduling capabilities at the OS level. For example, implementation of such a capability on the Linux OS, in a way that is transparent to the applications, may be found in [13], [15] and [17].

We consider the following problem whereby the users want to perform executions of a workflow application on the described Grid during a time interval in the future. The users want to have full control of the start time of the executions, i.e. the user is the one that initiates each time the execution of the workflow. Each workflow application consists of one or more services s from a given set of services S, and instances of the same service may reside in multiple hosts and machines within the same host. In order to execute a workflow application, a reservation request, denoted as r, must be made prior to its execution and therefore execution requests with no prior reservation are automatically rejected. A reservation r has to be specified in terms of start-time and finishing-time for availability of resources. At the time a new reservation request (r_{new}) arrives to the resource management service, each of the machines in the underlying Grid may already host a set of advance reservations $R = \{r_1, r_2, \dots, r_n\}$ that were accepted in the past. When a new reservation request arrives, the resource management service makes an evaluation of the hosts and presents the user with different ways of running the workflow application on probabilistic guarantees. The cost of the offers fluctuates according to the calculated by the resource management service probabilistic guarantee and the host's business policies. If the user chooses to accept one of the offers, the resource management service assigns the sub-services of the workflow application to the chosen machines within the hosts for future processing. For simplicity and without loss of generality, we assume that the completion time of a service is dominated by the execution time of the service itself, thus a possible delay that may be encountered when communicating with the hosts is considered negligible.

B. Assumptions

More formally, the problem may be defined as follows:

- Relatively to advance reservations, time is specified in time slices $\{T_k\}_{k\in\mathbb{N}}$ of X time units, following the paradigm of time slice based processors, with $T_{k+1} = T_k + X$. Each time slice represents the smallest temporal unit. Given $h \in \mathbb{N}$, a time horizon TH can be defined as $TH = \{T_k | k \in \mathbb{N} \wedge k \leq h\}$. Within TH, the user may reserve any time period TR defined as

$$TR = \{T_k \in TH | k \in \mathbb{N} \wedge start \leq k \leq end\} \quad (1)$$

and thus the duration of TR is $(end - start)X$.

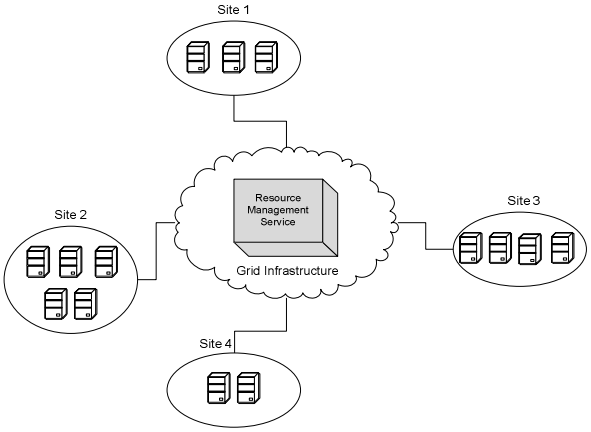

Figure 1. The Grid model.

- Each service $s \in S$, activated in the context of the reservation r, is associated with a worst-case execution time (WCET) ${C_{rs}}^m$ that the service may need on a given machine m (the machine index m will be omitted from the notation whenever implicitly identified). In the following, we assume that the WCETs $\{C_{rs}\}$ of the services have been estimated by assuming entire execution on a single CPU, among the available ones on a given machine. Also, for the sake of simplicity, we do not address the issue of how and whether to exploit parallelism on the underlying CPU, when dealing with a single service that needs to be run on a physical node (note that parallelism in a workflow is not completely ruled out).
- As already stated, users want to perform multiple runs of the workflow application without fixed execution start times. To accommodate this requirement imposed by the unknown number of executions and arbitrary start time, the resource management service evaluates and reserves computational resources for each service in the workflow across the same time interval, which is equal to the reserved time for the entire workflow TR_r. However, it is reasonable to assume that, within the time period reserved by the user for all services in the workflow, each service is active only for a short time within it.
- Each execution of a workflow application associated to a reservation r corresponds to one at most activation of each service s involved in the corresponding workflow (having a WCET of C_{rs}). In our model, we are using general service times (as opposed to exponential service times), and each reservation r is associated with a minimum inter-arrival period T_r, corresponding to the minimum time that may elapse between two consecutive requests of activation of the same workflow and consequently between two consecutive activations of the same service in the workflow. The reservation on underlying physical

resources should be tuned to sustain at most one execution of the entire workflow every time T_r.

- Each service s needed within each advance reservation r is associated with a deadline d_{rs} constituting a timing constraint: the execution of the service instance should only be allowed if the response time (i.e. the time between the activation and the completion of the service execution) is less than or equal to the deadline. More formally, if we denote the activation time of the service by a_{rs}^k (that may not necessarily be equal to the start time due to the scheduling of other activities) and the finishing time by f_{rs}^k, then this constraint may be formalized as:

$$\forall k, f_{rs}^k - a_{rs}^k \leq d_{rs} \quad (2)$$

- The response-time for the k^{th} activation of a service s, can be defined as $\rho_{rs}^k = f_{rs}^k - a_{rs}^k$. The minimum allowed value for ρ_{rs}^k is C_{rs}, but such a strict value, which is the best achievable one, would imply no possibility for time-sharing the same physical node with other activities.

4. Proposed Methodology

A. Grouping of overlapping reservations

Consider a reservation r that will host an application workflow that consists of services and is characterized by a QoS constraint expressed in terms of an end-to-end deadline D_r on the overall response-time ρ_r. We may assume that such a constraint is split into individual relative execution deadlines $\{d_{rs}\}$ for the services composing the workflow. For example, the technique in [19] may be used for such purpose.

From now on, we focus on a single machine m. Let R denote the set of reservations already allocated on m. We assume that the resource management service maintains a registry of such reservations. Using this registry, the time slices T_k on each machine can be grouped under common combinations of overlapping reservations. To formulate this process, we introduce the following notation:

- Let $G = \{g_1, g_2, \dots, g_v\}$ be the set of different groups of overlapping reservations in a given machine during TR_{new}. Each group g is a set, whose elements are the overlapping reservations. For each g, let rn_g be its cardinality ($rn_g \equiv |g|$), which indicates the number of included reservations within g.

- Let T_g and tn_g be the set of time slices T_k and their number respectably inside a given group $g \in G$.

To clarify this grouping process, we give the following example: let us consider the case of a candidate machine which has two pre-existing reservations r_1 and r_2 with reserved time intervals $T_3 \leq TR_1 \leq T_5$ and $T_5 \leq TR_2 \leq T_7$ respectably. We examine the arrival of a new advance reservation request, $r_{new} = r_3$with reserved time interval $T_2 \leq TR_3 \leq T_6$. For the requested service we discover candidate hosted machines. As candidate machines at this early stage, we consider all the machines m that are hosting the described service on which the worst-case execution time of service ${C_{rs}}^m$ doesn't exceed the deadline d_{rs} defined by the user.

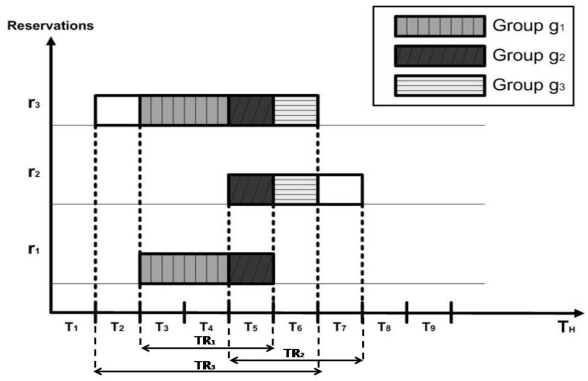

Figure 2. Grouping of time slices according to overlapping reservations.

Figure 2. depicts this situation and serves as a snapshot of the machine's "reservation" state across the time horizon. As illustrated, the introduction of the new reservation r_3 is translated into a set of groups G, which encapsulates three different groups, i.e. $G = \{g_1, g_2, g_3\}$. There is no pre-existing reservation during time slice T_2. Hence, T_2 in not included in any group since there is no conflict. On the other hand, during time slices T_3 and T_4 there is potential conflict between the new reservation r_3 and the pre-existing reservation r_1. Therefore, both these time slices are grouped under g_1, where $g_1 = \{r_1, r_3\}, T_{g_1} = \{T_3, T_4\}$ and so on. At the end of this process all time slices within the requested reserved time are grouped under common pre-existing reservations. This grouping process takes place for all candidate machines in the underlying Grid every time a new reservation arrives and helps to narrow down the complexity of the problem by avoiding identical calculations that take place during the next stages of the algorithm.

B. Probabilistic Model

As already mentioned, the end user is given the capability to start the execution of the service at any time within the reserved time interval. Having such a degree of flexibility of the execution start time, adds a significant degree of uncertainty to the advance reservation problem. In order to address this issue, we model this problem using the Bayesian Probability Theory and formulate it with the help of the grouping process described earlier. More formally, let us consider the event of having the new reservation r_{new} executing during a given time slice within the reserved time interval. Using Bayesian theory, the sample space Ω of this event can be divided into the following two independent events: B_1 and B_2, with $\bigcup_{i=1}^{2} B_i = \Omega$ and $P(B_i) \geq 0$, for i = 1,2, where B_1 is the event of having only r_{new} executing in T_k, i.e. having zero conflict, whereas B_2 is the event of having more than zero and less or equal to maximum overlapping reservations executing along with r_{new} in T_k.

For example, let us consider again the example depicted in Figure 2 and focus on time slice T_5. Then, B_1 is the event of having only r_3 executing, whereas B_2 is having r_3 executing along with r_1 or with r_2 or with both r_1 and r_2 during T_5. These two events are the same for all time slices that belong in the same group and thus this calculation needs to be performed only once per group and not for each time slice within the

reserved time interval. Furthermore, it should be noted that there is no need to calculate these probabilities for those time slices that are not included in any group and therefore there is no possibility of overlapping with other reservations.

Any given group $g \in G$ that derives from this grouping process described in Section 3.A can be divided into two subgroups: g_c, which contains the reservations whose services are executing in a given time slice whereas $g_{c'}$ contains the rest, i.e. those that are not executing:

$$g = g_c \cup g_{c'}. \tag{3}$$

Focusing on a single time slice T_k of X units belonging to a group g, let E_{rs} be the event of having the service s reserved for reservation r executing in the given time slice T_k. The arbitrariness of the start-time of each workflow instance is modeled as the knowledge of a probability π_{rs} that each service within a given reservation would actually be active in T_k if the underlying physical resource were utilized exclusively by it.

As we focus on a single machine m, for the sake of notational brevity, we will omit the dependency from s in sums and products of related terms, but we will maintain the subscript s in the notation, to stress that we refer to a service s inside a reservation r (deployed on m). Therefore, the probability $P(E_{rs})$ of the event E_{rs} can be written as:

$$P(E_{rs}) = \begin{cases} \pi_{rs} & \forall T_k \in TR \\ 0 & otherwise \end{cases} \tag{4}$$

For example, π_{rs} could be computed using queuing theory as the steady-state probabilities of each service having at least one item under processing, under an appropriate model for the stochastic process of the workflow activations. Using Eq. (3) and (4), the generic probability of having the new reservation grouped under common subgroup $g_c \subseteq g$ executing in any time slice, is given by the following generic equation:

$$P_{g_c} = P(B_2) = \prod_{r \in g_c} P(E_{rs}) \prod_{r \in g_{c'}} P(\overline{E_{rs}}) \tag{5}$$

where $\overline{E_{rs}}$ is the complementary event of E_{rs}, i.e. $P(\overline{E_{rs}}) = 1 - P(E_{rs})$. It is worth to note that, in Eq. (5), $P(E_{r_{new}s})$ is included in the first term of equation. When only the new reservation is executing, i.e. event B_1 (that is zero conflict), g_c includes only the new reservation $g_c = \{r_{new}\}$, whereas all the rest reservations in g are included in $g_{c'}$. Therefore, using Eq. (5) the probability of event B_1 is given by the following generic equation:

$$P(B_1) = P(E_{r_{new}s}) \prod_{r \in g \wedge r \neq r_{new}} P(\overline{E_{rs}}) \tag{6}$$

In the case of event B_2, the number of possible combinations of the reservations that overlap with r_{new} in a given group g_c is:

$$\frac{rn_g!}{i!(rn_g - i)!}, \text{ for all } 2 \leq i \leq rn_g \tag{7}$$

The probability of occurrence of each of these combinations P_{g_c} is calculated using Eq. (5). The results are the same for all time slices T_k grouped under the same $g_c \in g$, and this step needs to be performed only once per g. At the end of this stage, each group $g \in G$ is associated with the probabilities shown above.

C. *Real-time scheduling*

Assuming entire execution on a single CPU among the available ones on a given machine (see section 3.B) allows for the use of efficient partitioned processor scheduling strategies such as EDF. With this scheduling strategy, it is possible to reach theoretical full saturation of each processor, along with a very simple utilization-based admission control test that simply checks if the sum of the computation requirements for all of the active services on the same processor is less than or equal to 1:

$$\sum_{r \in R} \frac{C_{rs}}{min\{T_r, d_{rs}\}} \leq 1 \tag{8}$$

Also, by enriching the scheduling strategies with the mechanisms designed in the realm of the so called aperiodic servers, it is possible to provide the temporal isolation property, i.e., each reservation may receive scheduling guarantees independently of the behavior of other overlapping reservations. These algorithms are based on the assignment of a set of scheduling parameters: a budget Q_{rs} and a period P_{rs}. With such settings, the associated service: (a) is guaranteed the possibility to consume Q_{rs} time units of the resource in every period P_{rs}; (b) is forced not to overcome the consumption of Q_{rs} time units of the resource in each period P_{rs}.

Under these premises, and in order to fulfill a service deadline constraint, it is sufficient to tune the scheduling parameters for a service s within a reservation r by assigning a budget Q_{rs} equal to the estimated WCET C_{rs}, and minimum period P_{rs} equal to the minimum between T_{rs} and d_{rs}. As P_{rs} represents also the time granularity by which we may control the actual finishing time of each activation, it is useful also to set it to a sub-multiple of such quantity: $P_{rs} = min\{T_{rs}, d_{rs}\}/h_{rs}$, $Q_{rs} = C_{rs}/h_{rs}$, with $h_{rs} \in \mathbb{N}$, thus the reserved utilization $U_{rs} = \frac{C_{rs}}{min\{T_{rs}, d_{rs}\}} \equiv \frac{Q_{rs}}{P_{rs}}$ has not changed. This way, the admission control test, that checks whether or not the new reservation r_{new} may be admitted on the node under consideration, is given by:

$$\sum_{r \in R} U_{rs} + U_{r_{new}s} \leq U_{max} \tag{9}$$

where the parameter $U_{max} \leq 1$ has been introduced as the maximum capacity of a single processor, which may not necessarily be equal to 1. In fact, whenever scheduling overheads need to be accounted for, this value is set minor to 1, or the overhead has to be embedded in the WCET values.

Actually, as the potentially active reservations are not the entire set R but vary dynamically in time depending on what reservations are overlapping at any given time slice T_k, the just shown admission test needs to be repeated for each group of overlapping reservations as identified in Section 4.A. Consider the set G_r of groups in G including reservation r, $G_r = \{g \in G \mid r \in g\}$, the admission test for considering the machine under examination as a candidate becomes:

$$\forall g \in G_{r_{new}}, \sum_{r \in R} U_{rs} + U_{r_{new}s} \leq U_{max} \tag{10}$$

Finally, it is important to note that, under a scheduling reservation (Q_{rs}, P_{rs}), the maximum time needed by the service to compute once it is started may be approximated as:

$$\rho_{rs} = \left\lceil \frac{C_{rs}}{Q_{rs}} \right\rceil P_{rs} \cong \frac{C_{rs}}{U_{rs}} \tag{11}$$

where the index *k*, relative to each activation of the service, is omitted for simplicity.

D. *Taking advantage of arbitrary and independent service start-time*

The above shown assignment for resource scheduling parameters is adequate whenever the system is subjected to such a load that, within each reservation, each service works almost continuously, i.e., an actual usage of the reserved amount of resources for each reservation r that gets quite close to saturation. However, in the context of the problem formalized in Section 4.B, the services are not going to be continuously active within the time span TR_r due to their dependencies from other services in the workflow, or because the user do not simply use it. In such cases, it could be more convenient for the resource provider to exploit the expected statistical multiplexing among hosted applications (i.e., their activation patterns are supposed to be independent among each other) by offering a probabilistic guarantee at a lower rate, rather than a deterministic one at a higher rate. Of course, this would be translated in the opportunity to host within the same time slice a number of services that theoretically would not be allowed to do so, due to the processor's capacity.

By applying the real-time scheduling strategies described in section 4.C, service components have to share the underlying physical resource, so each service does not use it in an exclusive manner. For this reason, the probability $P(E_{rs})$ defined in Eq. 4 must be properly modified, taking into account the CPU share of the service. In virtue of the impact of the share U_{rs} on the response times of the service (see Eq. 11), it is reasonable to assume that the actual probability of having the service component actually active in each time slice be inversely proportional to the resource share, i.e.:

$$P(E_{rs}) = \begin{cases} \frac{\pi_{rs}}{U_{rs}} & \forall T_k \in TR \\ 0 & otherwise \end{cases} \tag{12}$$

Now, let us focus on a single time slice T_k, and the group g it belongs to. By following the reasoning in Section 4.B, for each subgroup g_c of g, $g_c \in \mathcal{P}(g)$ (where the $\mathcal{P}(\cdot)$ operator returns the possible subsets of its argument whose number is given by Eq. 7), it is possible to compute the probability P_{g_c} of having as active at the same exactly all of the reservations in g_c using Eq.5:

$$\forall g_c \in \mathcal{P}(g), P_{g_c} = \prod_{r \in g_c} \frac{\pi_{rs}}{U_{rs}} \cdot \prod_{r \in g_{c'}} \left(1 - \frac{\pi_{rs}}{U_{rs}}\right) \tag{13}$$

The available utilization by a new reservation can be considered as a discrete random variable $u(g_c)$. In fact, in the event of over-allocation, to comply with the time constraints of the existing reservations, this random variable may take a finite number of possible values, one for each subgroup of active overlapping reservations:

$$u_{g_c} = U_{max} - \sum_{r \in g_c \setminus \{r_{new}\}} U_{rs} \tag{14}$$

Therefore, the mean utilization U_O, related to the new reservation when experimenting resource overloads, is now given by the sum of possible values multiplied with the corresponding probability of the subgroups g_c that exhibit over-allocation. Denoting the set of these subgroups as g_{over}, with $g_{over} = \{g_c \in \mathcal{P}(g) \mid r \in g_c, \sum_{r,s} U_{rs} > U_{max}\}$, U_O can be obtained as:

$$U_O = \sum_{g_c \in g_{over}} u_{g_c} P_{g_c} \tag{15}$$

Likewise, denoting the complementary set of g_{over} as g_{over}', the mean utilization under zero overload (U_z) is given by the following equation:

$$U_z = U_{r_{new}s} \sum_{g_c \in g_{over}'} \left(1 - P_{g_c}\right) \tag{16}$$

Using the same reasoning and Eq. 11 it is also possible to estimate the expected response-time, conditioned to an activation of the workflow in any time slice T_k belonging to the group g:

$$E\left[\rho_{rs} | T_k \in T_g\right] = \sum_{g_c \in g_{over}} \frac{C_{rs}}{u_{g_c}} P_{g_c} + \frac{C_{rs}}{U_{r_{new}s}} \left(1 - \sum_{g_c \in g_{over}} P_{g_c}\right) \tag{17}$$

It is now possible to estimate the expected value of the overall execution response time for the new reservation during the time span $P_{rs} = min\{T_{rs}, d_{rs}\}$:

$$E[\rho_{rs}] = \frac{1}{P_{rs}} \sum_{g \in G} tn_g E\left[\rho_{rs} | T_k \in T_g\right] + (P_{rs} - \sum_{g \in G} tn_g) \frac{C_{rs}}{U_{r_{new}s}} \tag{18}$$

The value obtained for the expected response time could be used by the resource management service to check whether or not a given allocation of the services of the new reservation could be made to certain host machines, giving that its cost is acceptable by the user. For example, if D_{rnew} denotes the average end-to-end response time that the allocation should respect, then the test could be written as: $\sum_s E\left[\rho_{r_{new}s}\right] \leq D_{r_{new}}$. It should be noted that such a test could be easily incorporated as a QoS parameter inside the Service Level Agreement (SLA) established between the client and the provider prior to the reservation.

5. NUMERICAL EXAMPLE

As an example of the proposed technique, consider a time interval related to four specific machines inside which there are already three reservations $\{r_1, r_2, r_3\}$ allocated, with the parameters shown in the following table. For simplicity, we

assume the machines under study provides only one service and have the same characteristics, thus the relative service index *s* can be omitted from following analysis and the values of the following table apply to each of the four machines under study.

TABLE I. EXECUTION PARAMETERS

r	C_r	$P_{rs} = d_r$	U_r	$\frac{\pi_r}{U_r}$
r_1	10	100	0.10	0.05
r_2	60	120	0.50	0.20
r_3	35	100	0.35	0.15
r_4	30	120	0.25	0.10

We also assume that there is only one potential group of conflict $g = \{r_1, r_2, r_3\}$. The number of time slices under conflict tn_g has a different value depending on the machine:{1,10, 50, 100}. Assuming for simplicity $U_{max} = 1$, in order to maintain deterministic guarantees, the minimum acceptable utilization for the new reservation to be hosted in the machine would be 0.05, so r_4 would be rejected by all machines under examination. However, by taking into consideration the probabilities column and analysis in Section 3, we obtain 7 different subgroups of overlapping reservations:

$$\{\{r_1, r_4\}, \{r_2, r_4\}, \{r_3, r_4\}, \{r_1, r_2, r_4\}, \{r_1, r_3, r_4\}, \{r_2, r_3, r_4\}, \{r_1, r_2, r_3, r_4\}\} .$$

From these 7 subgroups only two exhibit over-allocation, that is $\{r_2, r_3, r_4\}$ and $\{r_1, r_2, r_3, r_4\}$. By applying the rules specified in equations (12) to (18) we obtain the values shown in the following table.

TABLE II. EXPECTED RESPONSE TIME, AND UTILIZATION FOR DIFFERENT NUMBER OF TIME SLICES

R_{g_c}	P_{g_c}	tn_g	$E[\rho_{rs}]$	dev %
$\{r_1, r_4\}$	0.00340	1	120.0025	0.208
$\{r_2, r_4\}$	0.01615	10	120.0250	2.083
$\{r_3, r_4\}$	0.01140	50	120.1333	11.108
$\{r_1, r_2, r_4\}$	0.00085	100	120.2500	20.833
$\{r_1, r_3, r_4\}$	0.00060			
$\{r_2, r_3, r_4\}$	0.00285	U_0	0.000435	
$\{r_1, r_2, r_3, r_4\}$	0.00015	U_z	0.008100	

The different number of time slices under conflict affects the average utilization, as expected. If one of the values may be considered as acceptable by the user, considering the type of hosted services and the cost, then the new service r_4 could be hosted on the chosen machine by relying on a probabilistic guarantee.

6. CONCLUSION

In this paper, a methodology to provide real-time guarantees in flexible advance reservation has been presented. In particular, it allows for provisioning both deterministic and probabilistic guarantees for advance reservation with no restrictions on the execution start time. The methodology relies on an estimation of the overall utilization of a physical host while conforming to the constraints imposed by previous reservations and allows for presenting alternatives to the user, for executing the workflow application under different probabilistic guarantees and cost. The illustrative examples showed that the principle behind the methodology is sound, allowing its adoption in real-time systems that seek to cost-effectively handle the execution of workflow applications across a large time horizon.

REFERENCES

[1] I. Foster, C. Kesselmana and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations, International Journal Supercomputer Applications, Vol. 15, No. 3, 2001.

[2] W. Leinberger and V. Kumar. Information Power Grid: The new frontier in parallel computing?, IEEE Concurrency, Vol. 7, No. 4, pp. 75-84, October-December 1999.

[3] U. Schwiegelshohn, P.Wieder and R. Yahyapour. Resource Management for Future Generation Grids, In Future Generation Grids, V. Getov, D. Laforenza, A. Reinefeld (Eds.), Springer, CoreGrid Series, 2005.

[4] S. McGough, A. Afzal, J. Darlington, N. Furmento, A. Mayer, and L. Young. Making the Grid Predictable through Reservations and Performance Modelling. The Computer Journal, 48(3):358-368, 2005.

[5] U. Farooq, S. Majumdar, E. W. Parsons. Efficiently Scheduling Advance Reservations in Grids, Technical Report, Dept. of Systems and Computer Engineering, Carleton University, Ottawa, Canada, August 2005.

[6] A. S. Netto, K. Bubendorfer and R. Buyya. SLA-based advance reservations with flexible and adaptive time QoS parameters, Proceedings of the International Conference on Service Oriented Computing, pp. 119--131, Vienna, Austria - September 17-20, 2007. Springer Verlag Lecture Notes in Computer Science.

[7] W. Smith, I. Foster and V. Taylor. Scheduling with Advanced Reservations, Proceedings of the IEEE/ACM 14th International Parallel and Distributed Processing Symposium, Cancun, Mexico, May 2000.

[8] H. Topcuoglu, S. Hariri, and M-Y Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans. on Parallel Distributed Systems. Vol 13, 3 (2002) 260-274.

[9] I. Stoica, H. Abdel-Wahab, K. Jeffay, and S.K. Baruah, J. E. Gehrke and C.G.Plaxton. A Proportional Share Resource Allocation Algorithm for Real-Time, Time-Shared Systems, Proceedings of the IEEE Real-Time Systems Symposium, Dec. 1996.

[10] S.K. Baruah, N.K. Cohen, C.G. Plaxton and D.A. Varve. Proportionate Progress: A Notion of Fairness in Resource Allocation, Algorithmica, 1996.

[11] C. W. Mercer, S. Savage, and H. Tokuda. Processor capacity reserves for multimedia operating systems. Technical Report CMU-CS-93-157, CarnegieMellon University, Pittsburg, May 1993.

[12] L. Abeni and G. Buttazzo. Integrating multimedia applications in hard real-time systems. In Proc. of the IEEE Real-Time Systems Symposium, Madrid, Spain, December 1998.

[13] L. Palopoli, T. Cucinotta, L. Marzario and G. Lipari. AQuoSA – adaptive quality of service architecture. Software – Practice and Experience, Vol. 39, n.1, pp. 1—31, 2009. Available at: http://dx.doi.org/10.1002/spe.883

[14] D. A. Lifka. ANL/IBM SP Scheduling System. Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing (IPPS '95), 1995.

[15] D. Faggioli, A. Mancina, F. Checconi and G. Lipari. Design and Implementation of a POSIX Compliant Sporadic Server for the Linux Kernel. 10th Real-Time Linux Workshop, Guadalajara, Mexico, 2008.

[16] C.L. Liu, J.W. Layland, Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment, Journal of the ACM, 20(1):46--61, 1979.

[17] G. Buttazzo, G. Lipari, L. Abeni, and M. Caccamo. Soft Real-Time Systems: Predictability vs. Efficiency, Springer, 2005.

[18] B. B. Brandenburg, A. D. Block, J. M. Calandrino, U. Devi, H. Leontyev, and J. H. Anderson. LITMUSRT: A Status Report, 9th Real-Time Linux Workshop, 2007.

[19] Y. Yuan, X. Li, Q. Wang and X. Zhu. Cost Optimization Method for Workflows with Deadline Constraints in Grids, Proceedings of the 2007 11th International Conference on Computer Supported Cooperative Work in Design.