Negotiation to Execute Continuous Long-Term Tasks

Valeriia Haberland and Simon Miles and Michael Luck

Abstract. Recently, research has focused on processing tasks that require continuous execution to produce data in a real-time manner. Such tasks often also need to be executed for long periods of time, such as years, requiring large amounts of resources (e.g. CPUs) that can be found in a Grid. However, a Grid may be unwilling or unable to allocate resources for continuous usage far in advance, because of high fluctuations in resource availability and/or resource demand. Therefore, a client must relax its requirements in terms of long-term usage, because resources could be required by other clients and cannot be monopolised by one client for as long as, e.g. one year. Such tasks often also need to be executed for long periods of time, requiring large amounts of resources (e.g. CPUs) that require continuous execution to produce data in a real-time manner. Some research seeks to find an optimal schedule for multiple continuous processes (e.g. continuous mixing of feedstock, packing) to utilise some resources (e.g. equipment) on plants [2], while other research addresses a dynamic area partitioning among a group of robots in order to monitor it continuously in respect of some events of interest (e.g. gas leaks) [1].

A Grid [4] potentially offers more resources (e.g. CPUs) than any supercomputer or cluster. However, a Grid might be unable or unwilling to schedule long-term tasks for the whole requested time duration, because resources could be required by other clients and cannot be monopolised by one client for as long as, e.g. one year. Such task will likely be allocated the execution time which is shorter than it was requested and this task will experience a planned interruption when this period ends. A client’s role in task execution increases when it negotiation is crucial by itself, e.g. for a continuous task’s interruption, because resource availability, where this maximum is estimated based on periodic patterns in resource availability fluctuations. ConTask also considers whether the current duration of interruption, as well as its cumulative duration (including all previous interruptions), is too long. In general, a client becomes more generous in negotiation when an interruption is too long.

2 FORMAL MODEL

This section provides a formal model for continuous execution of a task, including a client’s utility. The start time of task execution \( t^{\text{end}}_{\text{tot}} \) is when a client has submitted its first request for resources to a Grid, while the end time for the whole long-term duration of task execution is \( t^{\text{end}}_{\text{tot}} \). Each task may have several interruption(s) within \( t^{\text{tot}} = t^{\text{end}}_{\text{tot}} - t^{\text{str}}_{\text{tot}} \), which are not simultaneous for multiple tasks. Each adjoined pair \((\tau^{\text{int}}_{i,l}, \tau^{\text{exe}}_{i,l})\) of a single interruption \( \tau^{\text{int}}_{i,l} \) and execution \( \tau^{\text{exe}}_{i,l} \) periods of time for continuous task \( i \), where an interruption period of time precedes a corresponding execution period of time, has its counter \( l = 1, 2, 3, \ldots \) within \( t^{\text{tot}} \). Here, each \( \tau^{\text{int}}_{i,l} \) starts at \( t^{\text{str}}_{i,l} \) and ends at \( t^{\text{end}}_{i,l} \). A cumulative duration of interruption \( \tau^{\text{cum}}_{i,l} = \sum_{k=1}^{l} \tau^{\text{int}}_{i,k} \) is the sum of all interruption durations from the first until the \( l \)th prior to the execution period \( \tau^{\text{exe}}_{i,l} \), and it reflects success of the client’s strategy over total execution time.

The utility gained by a client for each execution period \( \tau^{\text{exe}}_{i,l} \) is negatively affected by two factors: a duration of preceding interruption \( \tau^{\text{int}}_{i,l} \), a cumulative duration of preceding interruptions \( \tau^{\text{cum}}_{i} \). Effectiveness function \( E_{i,l}(t) \) shows the success level of execution of task \( i \) as a process over time within each execution period \( \tau^{\text{exe}}_{i,l} \), assigning a value from the range \([0, 1]\) to each point in time \( t \). The effectiveness function increases linearly when the task is running (during \( \tau^{\text{exe}}_{i,l} \)) and does not change when the task is interrupted (during \( \tau^{\text{int}}_{i,l} \)). The length of interruptions affects the values of the effectiveness function during the following execution period.

The impact of the length of interruption on the following execution period is calculated with damping functions \( SI(\tau^{\text{int}}_{i,l}) \) and \( CI(\tau^{\text{cum}}_{i}) \) in the range \([0, 1]\) for a single and cumulative interruptions respectively. The values of these functions decrease during the interruption period towards zero. Consequently, these functions are presented as \( SI(\tau^{\text{int}}_{i,l}) = 1/\left(e^{(\tau^{\text{int}}_{i,l} - \tau^{\text{int}}_{\text{min}})/\epsilon_{\text{int}}} + 1\right) \) and \( CI(\tau^{\text{cum}}_{i}) = 1/\left(e^{(\tau^{\text{cum}}_{i,l} - \tau^{\text{cum}}_{\text{max}})/\epsilon_{\text{cum}}} + 1\right), \) where \( \tau^{\text{int}}_{\text{min}} \) and \( \tau^{\text{cum}}_{\text{max}} \) are the durations of single and cumulative interruptions after which an increment of client utility in respect of the following execution period decreases in more than a half of its possible value, compared to the case when an execution period was allocated at once.

3 King’s College London, United Kingdom, email: valeriia.kornich, simon.miles, michael.luck@kcl.ac.uk
after a task had been interrupted, while $\epsilon_{\text{int}}$ and $\epsilon_{\text{cum}}$ determine the speed of decrease of these functions around $\tau_{\text{int}}^{\text{max}}$ and $\tau_{\text{cum}}^{\text{max}}$.

The effectiveness function $E_{i,t}(t)$ increases in the range from the level of effectiveness $E_{i,j-1}(t_{\text{end}},j-1)$, achieved by a task before interruption $\tau_{\text{int}}^t$ at time $t_{\text{end}},j-1$, multiplied by the values of functions $SI(\tau_{\text{int}}^t)$ and $CI(\tau_{\text{cum}}^t)$, towards the largest possible effectiveness 1 at time $t_{\text{end}}$ in the time interval $t \in [t_{\text{str}},t_{\text{end}}]$, as below if $j \equiv l - 1$:

$$E_{i,t}(t) = \begin{cases} \left(1 - E_{i,j}(t_{\text{end}},j)\right) + E_{i,j}(t_{\text{end}},j) & \text{if } t \in [t_{\text{str}},t_{\text{end}}], \text{ and } j = l - 1; \\ \times SI(\tau_{\text{int}}^t) + CI(\tau_{\text{cum}}^t) & \text{if } t \in [t_{\text{end}},t_{\text{end}}], \text{ and } j = l - 1; \\ \end{cases}$$

(1)

Client utility $U_i$ for a task $i$ is the sum of all squares under the effectiveness function for the execution periods in proportion to the maximum possible square when a task does not have any interruptions, and is shown in Equation (2).

$$U_i = \frac{1}{\tau_{\text{tot}}} \sum_{l=1}^{l_i} \left(E_{i,l}(t_{\text{str}},l) + E_{i,l}(t_{\text{end}},l)\right) \times \tau_{\text{exe}},$$

(2)

where $l_i$ is the total number of interruption-execution pairs within $\tau_{\text{tot}}$, and the average utility $U_{\text{aver}} = \frac{1}{N} \sum_{i=1}^{N} U_i$ for $N$ tasks.

### 3 NEGOTIATION STRATEGY

An objective of our ConTask strategy for a client is to decrease the length of interruptions and increase the length of executions, while not losing significant utility. A client and GRA adopt an alternating proposals protocol [8], where the negotiators exchange proposals in turns, and both can either accept the opponent’s proposal or quit. In our case, the final deadline of negotiation is unknown, because a client must negotiate with the GRA until it obtains resources, but is time-dependent in terms of the length of interruption. Hence, we use time-dependent strategies [7] with nominally chosen deadlines, and negotiations repeat until an agreement is reached. Traditionally, an agreement is considered to be reached when the negotiator’s utility for the current opponent’s proposal is higher than for the next possible negotiator’s proposal. We propose to take a different approach when a client does not always accept a proposal which is more beneficial than its own next proposal, but may concede further by offering a shorter execution period to the GRA in order to end it at the approximate maximum resource availability, according to its periodic pattern. This decision reflects the continuity of task execution, because a client tends to reduce all possible interruptions within $\tau_{\text{tot}}$. The best offered execution period $\tau_{\text{exe}}^t$ is the last proposal of the GRA (or a client), which is to be accepted by a client or which is already accepted by the GRA, but has not yet been confirmed by a client. A client may offer a shorter $\tau_{\text{exe}}^t < \tau_{\text{exe}}^t$, but only on as long as to reach the closest maximum from the end of $\tau_{\text{exe}}^t$.

A client may also need to control the durations of interruptions if it is difficult to estimate the maximum resource availability. A client has to decide when single or cumulative interruptions are too long to reach an agreement faster. The values of dumping functions $SI(\tau_{\text{int}}^t(i))$ and $CI(\tau_{\text{cum}}^t(i))$ can be calculated during task’s interruption at time $t$ to ensure that their values have not significantly fallen towards 0. Assume that a client defines the sensitivity thresholds $\chi_{\text{int}}$ and $\chi_{\text{cum}}$ with respect to a single and cumulative interruptions respectively. As soon as the difference between the value of any

### Table 1. The average client utility for different negotiation strategies.

<table>
<thead>
<tr>
<th>Maximal deviation, %</th>
<th>ConTask, $\times 10^{-4}$</th>
<th>No ConTask, $\times 10^{-4}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9.93</td>
<td>4.46</td>
</tr>
<tr>
<td>10</td>
<td>6.80</td>
<td>3.49</td>
</tr>
<tr>
<td>50</td>
<td>4.25</td>
<td>1.67</td>
</tr>
</tbody>
</table>

This paper described our model of continuous long-term tasks execution in a Grid and, in general, our new ConTask negotiation strategy that allows near-contINUITY to be maintained among these tasks when they are running by increasing the duration of continuous running and by reducing planned interruptions.

### REFERENCES


