Safety scheduling strategies in distributed computing

Victor V. Toporkov*

Computer Science Department,
Moscow Power Engineering Institute,
ul. Krasnokazarmennaya 14,
Moscow, 111250 Russia
E-mail: ToporkovVV@mpei.ru
*Corresponding author

Alexey Tselishchev

European Organization for Nuclear Research (CERN),
1211 Geneva, 23, Switzerland
E-mail: Alexey.Tselishchev@cern.ch

Abstract: In this paper, we present an approach to safety scheduling in
distributed computing based on strategies of resource co-allocation for complex
sets of tasks (jobs). The necessity of guaranteed job execution until the time
limits requires taking into account the distributed environment dynamics,
namely, changes in the number of jobs for servicing, volumes of computations,
possible failures of processor nodes, etc. As a consequence, in the general case,
a set of versions of scheduling and resource co-allocation, or a strategy, is
required instead of a single version. Safety strategies are formed for structurally
different job models with various levels of task granularity and data replication
policies. We develop and consider scheduling strategies which combine
fine-grain and coarse-grain computations, multiple data replicas and
constrained data movement. These strategies are evaluated using simulations
studies and addressing a variety of metrics.

Keywords: distributed computing; scheduling; allocation; job; task; work.

Reference to this paper should be made as follows: Toporkov, V.V. and

Biographical notes: Victor V. Toporkov is a Professor and the Head of
Computer Science Department at Moscow Power Engineering Institute
(Technical University). He obtained his PhD and Doctor of Science in
Computing Systems and Networks Research from Moscow Power Engineering
Institute (MPEI). His primary research areas include distributed systems,
resource management and scheduling. He has published in journals such as the
Cybernetics and Systems Analysis, Journal of Computer and Systems Sciences
Int., Programming and Computer Software, Automation and Remote Control,
Automation Control and Computer Sciences.

Alexey Tselishchev is a Certification Authority Manager and an IT
Infrastructure Engineer at CERN, European Organization for Nuclear Research.
He obtained his Diploma in Computer Science (Dipl. Ing.) in 2006 at the
Technical University Ilmenau, Germany and obtained his Master of Science in
2008 at MPEI. His research interests are in the area of grid computing and
digital enterprise security.
1 Introduction

Scheduling of data processing in distributed environments including management information systems and real-time computing systems is enriched with a very essential component, namely, resource management (Czajkowski et al., 1998). The necessity of guaranteed job execution until time limits require taking into account the environment dynamics of the real-time system, namely, changes in the control object parameters, the number of demands for servicing, possible hardware failures, etc. Factors of real-time and control periods stipulate the generation of many different scenarios of scheduling and resource allocation basing on the information about the environment in which the given set of tasks is realised (Toporkov, 2005). This results in the notion of strategy, i.e., a set of supporting schedules of computations.

In some applications, complex sets of interrelated tasks (jobs) require co-scheduling (Ioannidou and Karatzas, 2006) and resource co-allocation (Czajkowski et al., 1999; Kurowski et al., 2003) in several processor (CPU) nodes. Each node may be in an autonomous administrative domain and be represented by a multiprocessor unit managed by a local batch management system, e.g., CODINE, LL, LSF, NQE, Condor, PBS, etc. (Skjellum et al., 2000). It is supposed that in local systems, resources are completely controlled, while it is not possible, for example, in the grid environment. In this case, resource allocation has a number of substantial specific features caused by autonomy, heterogeneity, dynamic changing of the contents and failures of nodes (Foster et al., 2001).

An analysis of the results of investigations on co-allocation of resources in distributed systems and grid makes it possible to conclude that job control can be efficient if realised on the basis of strategies (Hanzich et al., 2005; Ranganathan and Foster, 2002). This means a combination of different algorithms and scheduling heuristics (Tracy, 2001) with consideration for multiple factors and criteria: the policy of resource allocation and administration (Zhang et al., 2003), dynamical composition and heterogeneity of computing nodes (Krzhizhanovskaya and Korkhov, 2007), etc. Thus, it is necessary to consider the set of possible scenarios in computations. The choice of the strategy of resource co-allocation depends on such factors as the rate of load of CPU nodes (Thain and Livny, 2004) and the availability of data needed for computing; these factors lead one to use a flexible policy of data replication in the environment as well as their forward caching (Ranganathan and Foster, 2002; Tang, 2006). This includes also the need of using multicriteria models (Kurowski et al., 2003) taking into account the interests of users and resource owners, the resource-allocation policy adopted in a particular virtual grid organisation (Foster et al., 2001), as well as the mechanisms of its realisation (economic principles or quotas) (Buyya et al., 2002; Ernemann et al., 2002). The choice of resource composition and its reservation (Aida and Casanova, 2008; Ioannidou and Karatzas, 2006) depends also on the predicted time for the execution of a user program or on the availability of reference times stemming from the factor of real-time (Smith, 2003; Yang et al., 2003).

Methods for the generation of multicriteria strategies on the basis of a scalable scheduling model including for real-time computer systems were proposed and justified in earlier works (Toporkov, 2005, 2006). In the present paper, we propose multicriteria scheduling and resource co-allocation strategies for different job models. A strategy synthesis supposes that it is possible to generate a schedule of execution of a given set of tasks for each of the quality criteria, for example, the CPU load coefficients (Toporkov,
Safety scheduling strategies in distributed computing

2006). In the case of a single job model, it may occur that such a schedule does not exist. One possible reason is that there are no free CPUs because of failures in the system. Therefore, it is impossible to resolve the collisions of parallel tasks that compete for the same CPU node. Hence, it is necessary to have a set of strategies for job models which are structurally different and have various levels of granularity. We denote such sets of computing versions as safety strategies of scheduling and resource co-allocation.

The strategy is some kind of preparation of possible activities by a program-scheduler and its reaction to the events connected with resource assignment dynamics. The more factors considered as formalised criteria are taken into account in strategy generation, the more complete is the strategy in the sense of coverage of possible events. A specific variant (supporting schedule) should be selected from the strategy by the program-scheduler in accordance with the events occurring in the distributed environment and related primarily to the load, reallocation and forward reservations of resources.

To study the effectiveness of strategies with various task granularity and data replication policies, we have developed a simulation framework within which a particular schedule is chosen from the strategy by a metascheduler (Dail et al., 2003), on the analogy of an external scheduler (Ranganathan and Foster, 2002) and then sent to a local batch management system. The schedule depends on some control events occurrence in the distributed system (e.g., resource utilisation and workload), cost of job completion and CPU nodes performance. We use simulation results to evaluate three types of strategies addressing a variety of metrics, such as processor utilisation, average completion time, strategy time-to-live, cost of task completion and deviation of job reservation time from real-time of resource occupation.

2 Statement of the scheduling problem

Let \( T' \) be the set of job models associated with some totality of partially ordered tasks \( T = \{T_1, T_2, \ldots, T_n\} \) which should be executed before a limiting time \( t' \). The relation of the partial order on \( T \) is specified by a directed acyclic graph whose set of vertices corresponds to tasks of processing and remote data access. The set of arcs of the graph represents the informational and logical relations between the tasks. Figure 1 shows some examples of information graphs in strategies with different degrees of distribution, task details and data replication policies: the vertices \( T_1, T_2, T_3 \) and \( T_{43} \) correspond to processing, while \( d_1, d_2, d_3, d_4, d_5, d_6 \) and \( d_7, d_8 \) correspond to data transmission. The first strategy \( S_1 \), allows scheduling with fine-grain computations and multiple data replicas; the second strategy \( S_2 \), is one with fine-grain computations and a bounded number of data replicas; and the third strategy \( S_3 \), implies coarse-grain computations and constrained data replication. Task aggregating and diminishing the distribution degree is implemented, for instance, by transforming the graph for the strategy \( S_1 \) (see Figure 1) into the graphs for the strategies \( S_2 \) and \( S_3 \). The graph of the job model is parameterised by a priori (user’s) estimates, namely, the running time \( t_i' \) of task \( T_i \), \( i = 1, \ldots, n \), on the \( j \)th type of CPU resource; the cost of the resource employed; the amount \( v_j \) of computations on a CPU node of the \( j \)th type, etc.
The resource co-allocation $R$ for tasks in $T$ on a time interval $[0, t']$ is specified as follows:

$$R = \{(s_i, t_i, a_i)\}^{i=1,...,n}, \ a_i = j \lor j', \ j = 1,\ldots,J, \ j' \in \{1,\ldots,J\}, \ s_i \in [0,t'],$$

where $s_i$ and $t_i \geq t_i^0$ are the starting instant and running time of task $T_i \in T$, respectively, on a resource whose type is determined by the allocation $a_i$.

In equation (1), we have $a_i = j$ if task $T_i$ is assigned to a so-called basic processor resource whose level is bounded and depends on the distribution degree, the cost of the $j$th type resource and some other factors. If there is an occurrence of collision of tasks, which compete for the same resource of type $j$, then, we introduce a resource of type $j' \in \{1,\ldots,J\}$ whose performance characteristics are not worse than those of the basic resource and assign $a_i = j'$.

Depending on the aims in choosing the resource distribution, it may be an additional CPU of type $j$ or an original platform designed within the framework of the hardware/software codesign paradigm for embedded systems, when hardware and software components are concurrently developed along with task scheduling. The following problems are typical of hard real-time systems. Given deadlines for individual tasks and a job as a whole, minimise the cost function of hardware resources or maximise the basic CPU load and thereby minimise the cost of newly developed hardware platforms for a system.

The time allocated for the data transmission cannot be less than a minimal value determined by the capacity of the communication subsystem. If sequentially running
tasks are allocated to the same CPU, then the corresponding time of data exchange between these tasks is assumed to be zero. However, because there can be conflicts between the parallel processing tasks, the scheduling obtained before the exclusion of the exchange procedures cannot be revised (Toporkov, 2006).

We consider a scalable model of scheduling. The scalability means that the processor resource and memory can be upgraded and that the capacity of the communication subsystem is independent of the number of processor nodes involved in computations. Such a definition of this term has often been used in studies on parallel systems. There are well-known specimens of the scalable communication environments such as Myrinet, Memory Channel and communications based on the InfiniBand and RapidIO specifications. In most embedded systems, concurrent transmissions are not possible due to the usage of multicast and broadcast communication protocols. In this case, our communication model may be transformed by aggregating exchange procedures (e.g., by transforming the $S_1$ job graph into the graphs $S_2$ and $S_3$ in Figure 1).

Let us assume that we have the limitations to the time of execution of individual tasks and works as sequences $\{T_{i_1}, \ldots, T_{i_n}\}$ of informationally or logically related tasks, $1 \leq i_1 \leq \ldots \leq i_k \leq n$, that are components of the job:

$$t^*_g - t_g \geq 0, \quad t^*_h - \sum_{k=1}^{n} t_k \geq 0, \quad g, h \in \{1, \ldots, n\},$$

where $t_g$ and $t_h$ are the times of solution of tasks $T_g$ and $T_h \in T$, respectively, and $t^*_g$ and $t^*_h$ are the control times of termination of the task $T_g$ and the work involving the task $T_h$, respectively.

We represent a variant of admissible resource co-allocation and schedule $R$ in a quality criterion $w(r)$ by a vector $r = (t_1, \ldots, t_n, a_1, \ldots, a_n)$. We estimate the efficiency of the resource co-allocation (1) by the vector $W(r) = (w_1(r), \ldots, w_l(r))$, where $w_l(r)$, $l = 1, \ldots, L$, is a partial criterion. Let $w_j(\cdot)$ and $w_h(\cdot)$ be functions for estimating the efficiency of execution of the $i$th and $i_k$th tasks on resources with their types determined by the assignments $a_i$ and $a_k$, $k = 1, \ldots, K$. Let us represent the partial criterion $w(r)$ in the form:

$$w(r) = \sum_{i=1}^{n} w_i(t_i, a_i), \quad w(t_i, \ldots, t_k, a_i, \ldots, a_k) = \sum_{k=1}^{K} w_k(t_k, a_k) = \sum_{k=1}^{K} w_k(t_k).$$

An example of the additive criterion is the cost function for processing termination:

$$\text{CF} = \sum_{j=1}^{n} \left[ \frac{v_{ij}}{t_j} \right], \quad t_j \geq t^0_j,$$

where $v_{ij}$ is the relative volume of computations, $t_j$ is the time limit for the execution of task $T_j$ on a processor of the type $j$, $\left[ \frac{v_{ij}}{t_j} \right]$ is a partial function of cost of execution of
the \( i \)th task and \( \lceil \cdot \rceil \) denotes the nearest integer that is greater than or equal to a given value.

Let \( S \) be a strategy, i.e., a set of alternatives such that each alternative \( r \in S \) corresponds to an admissible resource co-allocation (1) under the constraints (2). The vector criterion \( W(r) \) containing partial criteria of the additive form (3) generates a binary relation \( F \) (e.g., the Pareto-relation) for comparison of alternatives on \( S \). We refer to the set of alternatives optimal with respect to \( F \) in the choice model \((S, F)\) (a special case of an algebraic system) as an \( F \)-optimal strategy of resource co-allocation. It is required to find an \( F \)-optimal strategy for job models of the set \( T_0^* \) corresponding to different levels of task granularity and data replication policies.

**Figure 2** Ranking critical works

All allocations to the processor node of the type 1

The 4th work

The 3rd work

The 2nd work

The 1st work

User’s time estimates

### 3 Method of critical works for job scheduling

By a critical work, we call the sequence of tasks with a maximum sum of user’s estimates for the duration when the best combination of resources is used, which contains the non-assigned tasks. The examples of parameters of processing tasks given in Table 1 correspond to the graphs \( S_1 \) and \( S_2 \) in Figure 1. User’s estimates for the duration \( \tau_{ij}^T \) of processing tasks \( T = \{T_1, \ldots, T_6\} \) and relative computing volumes \( v_{ij} \) for four types of base processor nodes are shown in Table 1, where \( i = 1, \ldots, 6; \ j = 1, \ldots, 4. \) When task aggregating, as shown in Figure 1, the values of the corresponding parameters of subtasks are summarised. Let us assume that the duration of all of the data exchanges for the graph \( S_1 \) is equal to one time unit, while data exchanges \( d_{12}, d_{78} \) in the strategies \( S_2, S_3 \) need two time units and data exchange \( d_{36} \) needs four time units. Let us rank the critical works in the graph \( S_1 \) in accordance with their prior maximum duration (Figure 2). For the ranked critical works allocated to the first CPU node, this value is equal to 12, 11, 10 and 9 units of time, respectively. In the graph \( S_1 \) (see Figure 1), the first critical work is matched by the sequence \((T_1, d_1, T_2, d_3, T_5, d_8, T_6)\) allocated to the CPU node of the first type. Its user’s maximum duration is equal to 12 units of time (see Table 1). After this sequence is assigned with tasks, the resources must be distributed for the following work \((T_1, d_1, T_2, d_4, T_5, d_8, T_6)\) because the tasks \( d_6, T_3 \) and \( d_8 \) are not
assigned and the prior duration of the work is 11 units (see Figure 2 and Table 1). The distribution of resources for the sequence \((d_{i_{4}}, T_{5}, d_{i_{8}})\) must take into account the results of assignment of tasks of the first critical work. The iterative application of this technique with a resolution of conflicts between tasks competing for one and the same resource constitutes the essence of the method of critical works based on dynamic programming (DP). This method makes it possible to construct a scheduling strategy for the partial criterion of efficiency of resource use.

| Table 1 | A priori estimates of task parameters in strategies \(S_1\) and \(S_2\) |
|-----------------------------------------------|
| **Duration, volume of computations** | **Task** |
| | \(T_1\) | \(T_2\) | \(T_3\) | \(T_4\) | \(T_5\) | \(T_6\) |
| \(i_{3}^0\) | 2 | 3 | 1 | 2 | 1 | 2 |
| \(i_{12}^0\) | 4 | 6 | 2 | 4 | 2 | 4 |
| \(i_{13}^0\) | 6 | 9 | 3 | 6 | 3 | 6 |
| \(i_{14}^0\) | 8 | 12 | 4 | 8 | 4 | 8 |
| \(\nu_{ij}\) | 20 | 30 | 10 | 20 | 10 | 20 |

Let \(w_{q}(r_{q})\) be a function for estimating the efficiency of the \(q\)th work with respect to criterion \(w(r)\), \(q = 1, \ldots, Q\), where \(Q\) is the number of works in the job, \(r_{q} = \{t_{1q_{1}}, \ldots, t_{nq_{1}}, a_{1q_{1}}, \ldots, a_{nq_{1}}\}\) is a variant of scheduling, \(n_{q}\) is the number of tasks, \(r = \{r_{1}, \ldots, r_{Q}\}\), and \(t_{iq_{j}}\) and \(a_{iq_{j}}\) are the duration and assignment of the \(iq_{j}\)th task, respectively, \(1 \leq i_{j} \leq n_{q}\). With regards to (3), the criterion \(w(r)\) can be represented as:

\[
w(r) = \sum_{q=1}^{Q} w_{q}(r_{q}) = \sum_{q=1}^{Q} \sum_{i_{j}=1}^{n_{q}} w_{iq_{j}}(t_{iq_{j}}, a_{iq_{j}}) = \sum_{q=1}^{Q} \sum_{i_{j}=1}^{n_{q}} \sum_{k=1}^{n_{q}} w_{iq_{j}}^{k}(z_{iq_{j}}) = \sum_{q=1}^{Q} w_{q}(t_{1q_{1}}, \ldots, t_{nq_{1}})
\]

The conditional extreme of the function \(w_{iq_{j}}(\cdot)\) in (5) for a given value of reserve \(z_{iq_{j}}\) by the time of execution at runtime of the \(iq_{j}\)th task can be obtained from the following DP functional equation:

\[
w_{iq_{j}}^{k} = \text{extr}_{j \in J} \left\{ w_{iq_{j}}(z_{iq_{j}}) \right\} + \sum_{k_{j} = i_{j} + 1}^{n_{q}} w_{iq_{j}}^{k}(z_{iq_{j}})
\]

where \(w_{iq_{j}}^{k}(z_{iq_{j}})\) is a conditional optimum of the function \(w_{iq_{j}}(t_{iq_{j}}, a_{iq_{j}})\) at a reserve \(z_{iq_{j}}\) by the time of execution at runtime of the \(k_{j}\)th task for the best combination of resource types.
The conditionally optimal values of the duration and assignment to a resource for a reserve \( z_{q_i} \) by time are determined from the following DP functional equations:

\[
\begin{align*}
    t_{q_i}^* \left( z_{q_i} \right) &= \arg \max_{t_{q_i} \in \Delta t_{q_i}} w_{q_i}^t, \\
    a_{q_i}^* \left( z_{q_i} \right) &= \arg \max_{j \in J} \left\{ w_{q_i}^t \left( t_{q_i}^* \left( z_{q_i} \right) \right), z_{q_i} \right\}.
\end{align*}
\]

In (7), \( \Delta t_{q_i} \) is the range of duration \( t_{q_i} \), depending on the reserve \( z_{q_i} \) by time at runtime of the corresponding task and \( t_{q_i}^* \left( z_{q_i} \right) \) in (8) is obtained from (7). In the case, when \( T_{q_i}, 1 \leq i \leq n \), is a part of the work \( \left( T_{q_i}, \ldots, T_{q_K} \right) \), \( 1 \leq i \leq \ldots \leq i_k \leq n \), the range for \( t_{q_i} \) is given in the following way:

\[
\Delta t_{q_i} = \left[ t_{q_i}^*, z_{q_i} - \sum_{g=1}^{K} t_{q_i}^* \right], \quad k = 1, \ldots, K-1, \quad \Delta t_{q_k} = \left[ t_{q_k}^*, z_{q_k} \right].
\]

In (9), \( t_{q_i}^* \) and \( t_{q_k}^* \) are the lower boundaries governed by the performance of the processor node in use (see Table 1) and \( z_{q_i} = z_{q_{i-1}} - t_{q_{i-1}} \), \( k > 1 \), represent the reserves by time at the start of execution of tasks \( T_{q_i}, T_{q_k} \).

Conditionally optimal is a resource co-allocation and schedule of the following form:

\[
r_{q_i}^* = \left( t_{q_i}^* \left( z_{q_i} \right), \ldots, t_{q_{n-q}}^* \left( z_{q_{n-q}} \right), a_{q_i}^* \left( z_{q_1} \right), \ldots, a_{q_{n-q}}^* \left( z_{q_{n-q}} \right) \right).
\]

The strategy \( S_q, q = 2, \ldots, Q \), is synthesised on the basis of the results of synthesis of a set of conditionally optimal \( S_{q-1}^* \) schedules (10), which is explained by the need of taking into account the information and logical links in the job and resolving conflicts between tasks competing for the use of a shared resource.

Let a partial criterion \( w(r) \) generate the binary relation:

\[
F^* = \left\{ (r', r^*) \bigg| r' \in S^*, \ S^* \subseteq S, \ r^* \in S \setminus S^* \right\}, w(r') \geq w(r^*),
\]

where \( S^* \) is the subset of schedules, being simultaneously both internal and external stable in the model of choice \( \left( S, F^* \right) \).

We will call \( S^* \) in (11) a strategy conditionally optimal with respect to the criterion \( w(r) \).

### 4 Single-criterion scheduling strategy: an example

Let us assume that we need to construct a conditionally optimal strategy of the distribution of processors according to equations (6)–(11) for a job represented by the
information graph $S_1$ (see Figure 1). The number of base CPU nodes of each type is equal to 1. The job termination time is given to be $t^* = 20$. The criterion of resource-use efficiency is a cost function of the form (4). It is required to construct a strategy [that is conditionally minimal with respect to (4)] for the upper and lower boundaries of the maximum range $\Delta t_i$ (9) for the duration $t_i$ of the execution of each task $T_i \in T$.

The conflicts between competing tasks of the set $C \subseteq T$ are resolved through unused nodes, which, being used as resources, are accompanied with a minimum value of the penalty cost function:

$$ CF^* = \sum_{c \in C} \left[ v_c / t_c^* \right], $$

(12)

where $v_c$ is the relative volume of computations on a processor node of the corresponding type, $t_c^*$ is the duration of execution of task $c$ if assigned to the base processor node, $t_c^* = t_0^*$ if the task $c$ is executed on an unused (to be used later) CPU node of the type $j \in \{1,...,J\}$; in this case, $t_0^*$ is a prior estimate of duration.

We choose the first critical work (see Figure 2) and apply to it functional equations (6)–(8) within the framework of the method of critical works. Let us introduce the following notations: $c_1, c_2, c_4$ and $c_6$ be partial functions of the cost of execution of the tasks entering into criterion (4); $CF_1$, $CF_2$ and $CF_4$ be the total costs of execution of the tasks $T_1, T_2, T_4, T_6$; $T_2, T_4, T_6$; and $T_4, T_6$; respectively. The maxima of variation ranges of the durations $t_1, t_2, t_4$ and $t_6$ in the corresponding tasks by (9) are:

- $\max \Delta t_1 = [2, 10]$;
- $\max \Delta t_2 = [3, 11]$;
- $\max \Delta t_4 = [2, 10]$; and
- $\max \Delta t_6 = [2, 10]$.

To find the conditional minimum $CF_1$, it is necessary, in line with (6), to search for minimum values of functions $CF_2$ and $CF_4$ depending on the reserve $z_2$ and $z_4$ at runtime of the task $T_2$ and $T_4$ (Figure 3). For $t_2^* \in \{2,10\}$, taking into account the time interval $t^*$ and duration of data exchanges, we obtain: $z_2 \in \{7,15\}$ and $z_4 \in \{4,12\}$. For $z_4 = 15, z_4 = 4$, we have:

- $t_4^* = 2$ and $t_4^* = 2$, the conditional minimum $CF_4^0 = c_4 + c_6 = 10 + 10 = 20$.
- For $z_4 = 12$, the conditional minimum $CF_4^0 = 12$ is found by two pairs of the values of $t_4^*$ and $t_6^*$: $t_4^* = 2, t_6^* = 2$ and $t_4^* = 10, t_6^* = 10$. The values of $CF_4$ include only $CF_4^0$. Indeed, if $z_2 = 7$ and $z_4 = 4$, the conditional minimum $CF_0^4 = 20 + 10 = 30$, with $t_4^* = 3$. For $z_2 = 15$ and $z_4 = 12$, we have: $CF_0^2 = CF_0^4 + c_2 = 12 + 10 = 22$, with $t_4^* = 3$ according to (7). Finally, the conditional optimum $CF_1 = c_1 + CF_0$ is reached at $t_1^* = 2, z_2 = 15$ and $t_1^* = 10, z_2 = 7$. In this case, $CF_1 = 32$ (see Figure 3). The assignments $a_1^*, a_2^*, a_4^*$ and $a_6^*$ are determined from (8). In (4), we take a prior estimate for the duration $t_0^*$ that is the nearest to $t_0$, which determines the type $j$ of the processor node used.
Figure 3  DP scheme for the first critical work scheduling

Figure 4  Scheduling critical works
<table>
<thead>
<tr>
<th>#</th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
<th>$t_4$</th>
<th>$t_5$</th>
<th>$t_6$</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
<th>$a_4$</th>
<th>$a_5$</th>
<th>$a_6$</th>
<th>CF</th>
<th>UP$_1$</th>
<th>UP$_2$</th>
<th>UP$_3$</th>
<th>UP$_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>41</td>
<td>0.35</td>
<td>0.10</td>
<td>0.15</td>
<td>0.50</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>10</td>
<td>10</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>37</td>
<td>0.35</td>
<td>0</td>
<td>0.65</td>
<td>0.50</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>41</td>
<td>0.35</td>
<td>0</td>
<td>0.15</td>
<td>0.50</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>41</td>
<td>0.35</td>
<td>0.10</td>
<td>0.15</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>10</td>
<td>10</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>38</td>
<td>0.35</td>
<td>0</td>
<td>0.15</td>
<td>0.50</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>11</td>
<td>11</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>39</td>
<td>0.35</td>
<td>0.10</td>
<td>0</td>
<td>0.55</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>39</td>
<td>0.35</td>
<td>0.10</td>
<td>0.15</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>11</td>
<td>11</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>39</td>
<td>0.35</td>
<td>0.10</td>
<td>0.15</td>
<td>0.55</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>28</td>
<td>0.35</td>
<td>0.10</td>
<td>0.15</td>
<td>0.50</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>11</td>
<td>11</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>41</td>
<td>0.35</td>
<td>0.10</td>
<td>0.55</td>
<td>0.55</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>41</td>
<td>0.35</td>
<td>0.10</td>
<td>0.60</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>41</td>
<td>0.35</td>
<td>0.10</td>
<td>0.15</td>
<td>0.50</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>10</td>
<td>10</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>39</td>
<td>0.35</td>
<td>0</td>
<td>0.65</td>
<td>0.50</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>41</td>
<td>0.35</td>
<td>0.10</td>
<td>0.15</td>
<td>0.50</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>39</td>
<td>0.40</td>
<td>0.20</td>
<td>0.15</td>
<td>0.20</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>41</td>
<td>0.40</td>
<td>0.30</td>
<td>0.15</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>40</td>
<td>0.40</td>
<td>0.20</td>
<td>0.30</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>43</td>
<td>0.35</td>
<td>0.35</td>
<td>0.15</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>43</td>
<td>0.60</td>
<td>0.10</td>
<td>0.15</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>39</td>
<td>0.60</td>
<td>0</td>
<td>0.15</td>
<td>0.20</td>
</tr>
<tr>
<td>21</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>41</td>
<td>0.60</td>
<td>0</td>
<td>0.15</td>
<td>0.25</td>
</tr>
<tr>
<td>22</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>42</td>
<td>0.60</td>
<td>0.30</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>41</td>
<td>0.60</td>
<td>0.10</td>
<td>0.15</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>40</td>
<td>0.60</td>
<td>0</td>
<td>0.30</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 2

Scheduling for the strategies $S_1, S_2$ (continued)

<table>
<thead>
<tr>
<th>#</th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
<th>$t_4$</th>
<th>$t_5$</th>
<th>$t_6$</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
<th>$a_4$</th>
<th>$a_5$</th>
<th>$a_6$</th>
<th>$CF$</th>
<th>$UP_1$</th>
<th>$UP_2$</th>
<th>$UP_3$</th>
<th>$UP_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>41</td>
<td>0.60</td>
<td>0.10</td>
<td>0</td>
<td>0.20</td>
</tr>
<tr>
<td>26</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>43</td>
<td>0.60</td>
<td>0.10</td>
<td>0.15</td>
<td>0</td>
</tr>
<tr>
<td>27</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>43</td>
<td>0.30</td>
<td>0.40</td>
<td>0</td>
<td>0.30</td>
</tr>
<tr>
<td>28</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>41</td>
<td>0.40</td>
<td>0.45</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>29</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>40</td>
<td>0.40</td>
<td>0.50</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>41</td>
<td>0.40</td>
<td>0.50</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>31</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>43</td>
<td>0.35</td>
<td>0.50</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>32</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>43</td>
<td>0.35</td>
<td>0.35</td>
<td>0.15</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>39</td>
<td>0.40</td>
<td>0.20</td>
<td>0.35</td>
<td>0</td>
</tr>
<tr>
<td>34</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>40</td>
<td>0.35</td>
<td>0.25</td>
<td>0.40</td>
<td>0</td>
</tr>
<tr>
<td>35</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>42</td>
<td>0.30</td>
<td>0.40</td>
<td>0.30</td>
<td>0</td>
</tr>
<tr>
<td>36</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>41</td>
<td>0.40</td>
<td>0.30</td>
<td>0.15</td>
</tr>
<tr>
<td>37</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>40</td>
<td>0.40</td>
<td>0.20</td>
<td>0.30</td>
<td>0</td>
</tr>
<tr>
<td>38</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>41</td>
<td>0.40</td>
<td>0.30</td>
<td>0.20</td>
<td>0</td>
</tr>
<tr>
<td>39</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>43</td>
<td>0.35</td>
<td>0.35</td>
<td>0.15</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>43</td>
<td>0.35</td>
<td>0.35</td>
<td>0.15</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>39</td>
<td>0.40</td>
<td>0.20</td>
<td>0.15</td>
<td>0.20</td>
</tr>
<tr>
<td>42</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>40</td>
<td>0.35</td>
<td>0.25</td>
<td>0.15</td>
<td>0.25</td>
</tr>
<tr>
<td>43</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>42</td>
<td>0.30</td>
<td>0.40</td>
<td>0</td>
<td>0.30</td>
</tr>
<tr>
<td>44</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>41</td>
<td>0.40</td>
<td>0.30</td>
<td>0.15</td>
<td>0</td>
</tr>
<tr>
<td>45</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>40</td>
<td>0.40</td>
<td>0.20</td>
<td>0.30</td>
<td>0</td>
</tr>
<tr>
<td>46</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>41</td>
<td>0.40</td>
<td>0.30</td>
<td>0</td>
<td>0.20</td>
</tr>
<tr>
<td>47</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>43</td>
<td>0.35</td>
<td>0.35</td>
<td>0.5</td>
<td>0</td>
</tr>
</tbody>
</table>
Then, we distribute the resources for the sequence of tasks \( (d_4, T_4, d_8) \), \( (d_5, T_5, d_6) \) and \( (d_6) \), entering into the second, third and fourth critical works, respectively (see Figure 2). The strategy with a conditional minimum with respect to CF (4) is shown in Table 2 and Figure 4 as co-allocations and schedules 1, 2 and 3. The conflict between the tasks \( T_3 \) and \( T_5 \) in schedule 2 is resolved by assigning the task \( T_3 \) to a CPU node of type 3 and the task \( T_5 \) to a node of type 4. In so doing, the minimum \( CF^* = 7 \) in (12) is ensured.

5 Multicriteria and multimodel scheduling strategies

Let us have a job model from \( T_0^* \) used in functional equations (5)–(10) for generating a strategy \( S_l^* \) that is conditionally optimal with respect to criterion \( w_l(r) \) entering into the vector \( W(r) \). By analogy with (11), the criterion \( w_l(r) \) generates a partial binary relation:

\[
F_l = \left\{ (r', r^*) \mid r' \in S_l^*, S \subseteq S, r^* \in S \setminus S_l^* \right\}, \quad w_l(r') \geq w_l(r^*),
\]

where the strategy \( S_l^* \) is internally and externally stable in the model of choice \((S, F_l)\). The generation of \( S_l^* \), \( l = 1, \ldots, L \), can be performed for all criteria entering into \( W(r) \) in line with the method of critical works.

Let us consider an application of the method of critical works for the vector criterion to the job model of the strategy \( S_1 \) (see Figure 1). Let the relation \( F \) in \((S, F)\) have the form of:

\[
r'F r^* \iff w_l(r') \geq w_l(r^*), \quad r', r^* \in S, \quad l = 1, \ldots, L
\]

The vector criterion includes the cost function CF (4) and load coefficients \( UP_j \), \( j = 1, \ldots, 4 \), of base CPU nodes. Since there are no conditional branches in the job graph (see Figure 1), \( UP_j \) is the ratio of the total time of usage of a processor node of the type \( j \) to the time interval \( t \) of the job completion. The strategies that are conditionally maximal with respect to criteria \( UP_1, UP_2, UP_3 \) and \( UP_4 \) are given in Table 2 by the cases 4–7, 8, 9, 10, 11; and 12–14, respectively. The \( F \)-optimal strategy includes all 14 schedules (the cases 1–3 present the strategy for CF). This result is consistent with (13), (14). Let us assume that the load of CPU nodes is such that the tasks \( T_3, \ldots, T_5 \) can be assigned with no more than three units of time on the first and third nodes (see Table 2).

The metascheduler runs through the set of schedules and chooses a concrete variant of resource distribution that depends on the actual load of processor nodes. Then, the metascheduler should choose the schedules 1, 2, 4, 5, 12 and 13 as possible variants of resource distribution. However, the concrete schedule should be formulated as a resource request and implemented by the batch management system subject to the state of all four processor nodes and possible runtimes of tasks \( T_4, \ldots, T_6 \) (see Table 2).

Let us assume that one needs to generate a Pareto-optimal strategy of CPU nodes distribution for the job model in the strategy \( S_2 \) (see Figure 1) and prior estimates in Table 1. The Pareto relation is generated by the vector of criteria CF, \( UP_1, \ldots, UP_4 \). The strategy is constructed in the whole range (9) of the duration \( t_i \) of solution of each task,
while the step of change is taken to be no less than the lower boundary of the range (9). The remaining initial conditions are the same as in the previous examples. The strategies that are conditionally optimal with respect to the criteria CF, UP, UP2, UP3 and UP4 are presented in Table 2 by the schedules 15–18, 19–26, 27–31, 32–39 and 40–47, respectively. The Pareto optimal strategy does not include the schedules 16, 19, 26, 28, 30, 31, 36 and 44.

Now, let us consider the application of the method of critical works to the set $T_0^*$ of job models represented structurally by the graphs $S_1$, $S_2$ and $S_3$ in Figure 1 and by the estimates given in Table 1. For $S_1$, the initial conditions are the same as in the example of Section 4 and, for $S_2$ and $S_3$, the conditions of the examples of the same section are adopted. It is required to construct an $F$-optimal strategy, where $F = \bigcup_{i=1}^{13} F_i$ and $F_i$ is generated by one of the criteria CF, UP, UP2, UP3, UP4. As a result of the resource distribution for the model $S_3$, the tasks $T_1$, $T_23$, $T_45$ and $T_6$ turn out to be assigned to one and the same processor node of the first type. Consequently, the data exchanges $d_{12}$, $d_{36}$ and $d_{78}$ can be excluded. Because there can be no conflicts in this case between processing tasks (see Figure 1), the scheduling obtained before the exclusion of exchange procedures can be revised.

The results of distribution of processors are presented in Table 3. Schedules 1–6 in Table 3 correspond to the strategy that is conditionally minimal with respect to CF with $UP_1 = 1$. Consequently, there is no sense in generating conditionally maximal schedules with respect to criteria $UP_2$, $UP_3$, $UP_4$. Thus, the $F$-optimal strategy coincides with the strategy presented in Tables 2, 3 for the job models in $S_1$, $S_2$ and $S_3$ (see Figure 1 and Table 1) accurate to the equivalence relation $\mathcal{F}$, where $\mathcal{F}$ is a quasi-order on $S$.

### Table 3 Scheduling for the strategy $S_3$

<table>
<thead>
<tr>
<th>#</th>
<th>$t_1$</th>
<th>$t_{23}$</th>
<th>$t_{45}$</th>
<th>$t_{6}$</th>
<th>$a_{1}$</th>
<th>$a_{23}$</th>
<th>$a_{45}$</th>
<th>$a_{6}$</th>
<th>$CF$</th>
<th>$UP_1$</th>
<th>$UP_2$</th>
<th>$UP_3$</th>
<th>$UP_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>8</td>
<td>6</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>25</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>8</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>24</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>4</td>
<td>6</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>24</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>27</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>29</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>11</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>32</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### 6 Simulation results and comparison between strategies

To study and to evaluate the effectiveness of scheduling strategies we have developed a metascheduler simulator. The simulator allows to generate and to choose a schedule as a resource reservation sent to a local job management system (Aida and Casanova, 2008; Ioannidou and Karatza, 2006). We have studied three types of strategies $S_1$, $S_2$ and $S_3$ for
more than 3,000 randomly generated jobs under the constraints described in Sections 2 and 3. CPU nodes were divided into four groups $N_1$, $N_2$, $N_3$ and $N_4$ in accordance to their performance. The relative performance of $N_1$ is equal to 1, the performance of $N_2$, $N_3$ and $N_4$ is equal to 0.5, 0.33 and 0.25, respectively. A number of CPU nodes were varied from 20 to 30. Running time and amounts of computations for different tasks of the job were selected randomly with a uniform distribution. These parameters for various tasks had difference which was equal to 2…3. Job parameters are presented by Table 4. For each simulation experiment, we measured an average CPU node utilisation [Figure 5(a)], task completion time and cost in the form (4) [Figure 5(b)], time-to-live of the strategy, i.e., a time interval at which scheduling is applicable taking into account dynamics of the distributed environment, and deviation of job reservation time from real-time of resource occupation [Figure 5(c)].

Table 4  Jobs parameters in diverse strategies

<table>
<thead>
<tr>
<th>Job parameters</th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average number of tasks</td>
<td>26.9</td>
<td>25.2</td>
<td>10.0</td>
</tr>
<tr>
<td>Average number of works</td>
<td>33.4</td>
<td>29.0</td>
<td>13.1</td>
</tr>
<tr>
<td>Average number of works with more than three tasks</td>
<td>6.2</td>
<td>5.3</td>
<td>2.3</td>
</tr>
<tr>
<td>Maximum length of works (in tasks)</td>
<td>5.3</td>
<td>4.4</td>
<td>3.7</td>
</tr>
</tbody>
</table>

The strategy $S_1$ is less complete than the strategy $S_2$ in the sense of coverage of events in distributed environment, because it is constructed for the best- and worst execution time estimations. However, the important point is the generation of a strategy by efficient and economic computational procedures of the program-scheduler. The type $S_2$ has more computational expenses than $S_1$.

The strategy $S_1$ performs the best in the term of load balancing for different groups of CPU nodes, while the strategy $S_2$ and especially the strategy $S_3$, tries to occupy nodes with the highest performance [see Figure 5(a)].

The strategy $S_1$ is remarkably worse than $S_1$, $S_2$ with respect to task completion time [see Figure 5(b)] because tasks in $S_1$ are presented by coarse-grain computations. Strategy $S_2$ is better than strategy $S_1$ with respect to task running time because $S_2$ is constructed on the whole interval of the variation of each running time $t_i$ and $S_1$ is synthesised for the upper and lower bounds of the interval (9) of the variation of the execution time $t_i$ (see Table 2). Thus, the schedules in $S_2$ are more accurate than the schedules in $S_1$.

The slow strategy $S_3$ is the cheapest one [see Figure 5(b)] and the costs of $S_1$, $S_2$ are equal in fact. The strategy $S_1$ is the best in the term of liveability and the fast strategy $S_2$ has the least time-to-live [see Figure 5(c)]. This may be explained as follows: the strategies of the type $S_1$ in all cases try to monopolise CPU resources with the highest performance and to minimise data exchanges (see Table 3).

The schedules in the strategies $S_1$ and $S_2$ have remarkably worse deviation of expected starting time from real resource occupation time than the schedules in $S_2$. The type $S_2$ is more accurate but schedules in $S_2$ are the worst with respect to time-to-live.
Figure 5  Relative characteristics of different strategies

7 Conclusions and future work

In this paper, we propose and compare different types of safety scheduling and resource co-allocation strategies in distributed computing systems. We consider computational models with different levels of task granularity and data replications.

Our results show, first of all, that the choice of scheduling strategy has a significant impact on distributed system performance and its utilisation. Second, we find out that it is important to address both task granularity and data replication policies. These results are promising, but we have bear in mind that they are based on simplified computation scenarios, e.g., in our experiments we use first-come-first-served (FCFS) management.
policy in local batch systems. In future work, we plan to explore such algorithms as the backfilling and the least-work-first and to study adaptive algorithms for resource selection.

Acknowledgements

This work was supported by the Russian Foundation for Basic Research (Grant No. 09-01-00095) and by the State Analytical Program ‘The Higher School Scientific Potential Development’ (Project No. 2.1.2/6718).

References


