Distributed Resource Management in Workflow Environments

Weimin Du
Hewlett-Packard Labs.
California, USA
du@hpl.hp.com

Graham Eddy
Hewlett-Packard
Melbourne, Australia
graham_eddy@hp.com

Jim Davis
Hewlett-Packard Labs.
California, USA
davis@hpl.hp.com

Ming-Chien Shun
Hewlett-Packard Labs.
California, USA
shan@hpl.hp.com

Abstract
Workflow processes involve the coordinated execution of tasks performed by (local or remote) workflow resources (e.g., a person, a computer-based application, a printer). Workflow resource management is concerned with keeping track of resource status and assigning available resources to workflow tasks. Traditional approaches either manage distributed resources globally at a central site incurring huge overhead to keep track of resource status, or manage resources locally resulting in long delays at run time resource assignment. In this paper, we describe a distributed resource management approach that addresses above problems. We introduce the concept of two-step resource assignment. Instead of doing resource assignment in one step either at a central site (in global management approach) or at remote sites (in local management approach), the proposed approach first checks at central site the availability of resource groups, and then selects at remote sites specific resources from the group. A resource group is a group of resources that all can perform certain tasks. It is much easier to keep track of the status of resource groups (e.g., at least some printers are available at a site) than that of individual resources. In the paper, we discuss specification of resource group and their relationships to individual resources. We also discuss propagation of individual resource status to resource groups at remote sites. We describe such an implementation in the context of HP OpenPM workflow management system.

Keywords Data replication, interdependent data, workflow systems, workflow resource management


1 Introduction
Workflow is a technology that provides the ability to define and automate the flow of work through an organization to accomplish business tasks. Business tasks are first modeled as workflow processes, which are then automated by workflow management systems (WFMSs).

As defined by Workflow Management Coalition [7], a workflow process is a coordinated set of process activities that are connected in order to achieve a common business goal. A process activity is a logical step or description of a piece of work (termed work item) that contributes toward the accomplishment of a workflow process. A work item defines the work to be processed in the context of a process activity and is performed by a workflow participant. A process activity may be a manual process activity and/or an automated workflow process activity and a workflow participant may be a person that performs work items for manual process activities or a computer-based application that performs work items for automated workflow process activities.

A workflow participant can usually perform work items for more than one process activity. The collection of process activities that can be assumed and performed by a workflow participant is termed a process role. In general a workflow participant can assume several process roles and a process role can also be assumed by several participants.

A participant, when performing a work item, may also need to use or access other objects. For example, a person who prints a document apparently needs a printer. Workflow participants, together with objects they use, are external resources a WFMS needs to execute a workflow process.

1.1 The Problem
The problem of resource management in WFMSs is concerned with efficiently assigning resources to
process activities at process execution time. We assume that there is a resource management subsystem (or resource manager) in each WFMS that does the following:

1. Keeping track of resource status; and
2. Finding eligible, available, and hopefully least loaded resources for activities when needed.

Resource management is easy if all resources are local and thus can be managed by a centralized resource manager. The problem becomes difficult when resources are distributed as in many workflow environments. Distributed resources can be managed in the following two straightforward approaches.

Global management approach

The global management approach manages distributed resources using a global resource manager (GRM). All resources will be registered to the GRM with roles they may assume. The GRM is also responsible for keeping track of status of registered resources.

Local management approach

In this approach, resources are managed by multiple, distributed local resource managers (LRMs). Each LRM has all status information of and full control over resources at its site. There may be a GRM at central site which maintains only role information of each resources, as well as their managing LRMs. The WFMS relies on individual LRMs for resource assignment when a work item is to be performed.

The main advantage of global management approach is that resource assignment is easy and efficient, as all resource information is local. However it incurs huge overhead in keeping track of status of remote resources. As described, the approach is not practical in real workflow environments for two reasons. First, the number of remote resources is usually very large. It is extremely difficult for the GRM to keep track of load information about remote resources as it changes frequently. Second, resources usually belong to different organizations. The GRM may not have the authority to assign some of the resources.

Local management approach avoids the huge overhead of keeping track of dynamic changes of resources by managing them locally. It however makes run time resource assignment difficult and inefficient. The GRM has to contact remote LRMs for resource status.

Most workflow products (e.g., IBM FlowMark [3]) adopt a simplified version of the above approaches. As in local management approach, the GRM maintains only static information of resources, e.g., roles and accessing addresses. There is however no LRMs that maintain dynamic information such as availability and work loads. It is therefore possible (and often the case) that a work item is assigned to resources that are not available at the moment or already heavily loaded. When this happens, the tasks will be queued, resulting long execution delay.

Efficient and reliable resource assignment is important for workflow process execution. This is especially true when organizations are re-engineering their businesses to rely on workflow systems for their business critical tasks. On the other hand, the problem is not easy, due to the large number of resources owned by different groups that have to be managed and their dynamic behavior.

The purpose of this paper is to address this issue. We realize that it is generally impossible to guarantee optimal resource assignments, as resource status changes all the time, even at resource assignment time. What we will describe in the paper are techniques that can significantly reduce operation overhead that is needed to maintain dynamic resource status without introducing long delay in run time resource assignment.

1.2 Distributed Resource Management

As we mentioned, there are two reasons why resource management is difficult:

- The number of resources is large.
- Resource status (e.g., availability and load) changes frequently.
- Resources belong to different organizations.

The paper proposes a distributed resource management approach that alleviates the above difficulties to some extent. Figure 1 shows the architecture of a distributed resource management system. As in local management approach, there is one GRM connected to the WFMS and a number of LRMs each manage a number of resources at their sites. Before we go into details of the approach, let us first introduce the concepts of resource groups and two-step resource assignment.

A resource group, as its name suggests, is a group of resources that all can assume a process role. For example, Allan, Bob, Charles, and Doug are all software engineers and thus form a resource group that can assume software engineer role. Resources in a resource group should be in the same site and managed by the same resource manager. A resource can be in one or more resource groups and a resource group may assume one or more process roles.

The basic idea of two-step resource assignment is to manage resources at two levels: resource
groups at the GRMs and individual resources at LRM. More specifically, LRM has information of and full control over each individual resource they manage. They keep track of not only static information of resources such as roles and addresses, but also dynamic status information such as availability and work load. The GRM, on the other hand, only has information about resource groups. The resource assignment is done in two steps.

1. The GRM first maps process role of a process activity to a resource group and checks the availability and load of the group. The LRM that manages the selected group is then contacted.

2. The LRM further maps the group into individual resources and checks their availability and work loads. The selected resources will then be informed or invoked to perform the work item.

The two-step resource management and assignment is motivated by the following two observations. The first observation is that, in most workflow environments, a process role can often be assumed by any one of several resources. The number of resource group is thus much smaller than that of individual resources. The second observation is that, although the status of an individual resource may change frequently, the status of a resource group changes much less frequently. For example, it is very likely that a particular person is on vacation at a given time, but it is very unlikely that all persons in a group are on vacation at the same time.

The advantages for having the GRM manage remote resource at the group level are two-fold. First, it reduces the number of individual resources it has to manage, due to the smaller number of resource groups. Second, and more importantly, it reduces the overhead of keeping track of remote resource status. In most cases, it is sufficient for the GRM to know that at least one resource that can perform the task is available and it is much easier to keep this information up to date than to keep track of status of every individual resources that can perform the task. With proper resource group definition, it is possible that the status of individual resources in a group changed many times before that of the group has been affected. By separating responsibilities between the GRM and LRM, it significantly reduces the number of updates that the GRM has to make in order to keep resource status in sync at the central site.

It worths noting that the traditional local management approach is just a special case of the proposed distributed management approach. This is true as each resource group can be a singleton. In general, the singleton resource group can also be used for important remote resources that need to be monitored individually by the GRM.

In the rest of the paper, we will describe details of the proposed distributed resource management approach. In particular, we will discuss specification of resource group and their dependency upon individual resources. We also discuss the propagation of status change of individual resources to resource groups. A possible implementation of the approach will be described in the context of HP OpenPM workflow management system [2].

1.3 Related Work

Workflow resource management can be considered as a special case of a more general problem of managing interdependent data maintained in different information systems [1] [5] [6] [8].

The general interdependent data management problem has previously been studied for data warehouse and multidatabase systems. Existing solutions are good for environments where source data do not change very often and for applications that can tolerate certain degrees of inconsistency between source and target data. The foci are therefore on consistency requirements such as specification of tolerable inconsistencies and ways to meet these (relaxed) consistency requirements. For example, the notion of eventual consistency has been proposed which guarantees that changes of source data will eventually be propagated to target data at specified time or in a specified time period.

The contributions of this work are not only on applying general interdependent data management techniques to workflow resource management problem, but also, and more importantly, on using semantics of workflow resource management to reduce overhead for keeping status of remote resource in sync. We focus on specification of resource groups and their dependency to individual resources, rather than on specification of tolerable inconsistencies as in
general approaches. Dependency between resource groups and individual resources are enforced not only using database techniques as described in previous work, but also using workflow techniques that are more flexible and powerful.

2 Resource Specification

A resource group is a three element tuple:

$$G = < S, C, A >,$$

where $S$ is the definition (including status specification), $C$ is a Boolean-valued consistency predicate that specifies when $G$'s status should be reevaluated, and $A$ is a collection of status update procedures that are used to bring $G$'s status up to date.

One objective of workflow resource management is to minimize the number of remote updates that invoke status update procedures to bring resource group status at remote sites up to date. The traditional approach is eventual consistency which delays propagations of resource status change to remote sites. The approach is unfortunately unacceptable as WFMSs requires up to date resource information. We address the problem by macro resource management which manages resource at group level. The approach is possible only when resource assignment is done in two steps as we described.

There are two issues in the proposed approach: the specification of resources, resource groups and their relationships which is described in the section, and the propagation of individual resource status changes to remote GRMs which is described in the next section.

2.1 Resources and Resource Groups

A resource group $G$ is modeled as an object with five attributes: name, members, capabilities, status and history.

Each resource group has a name and is characterized by a set of member resources. Capabilities, status, and history are the three attributes based on which the GRM assigns resource groups to work items. The capabilities of a resource group is used to determine the role it can assume, while the status indicates its availability and the current load. Since the resource manager does not have the exact status information of managed resources, it also uses history information of previous assignments as a hint of accuracy of recorded status data. Example history data include number of times a resource has been assigned a work item but is not available, and average waiting time of previously assigned work items.

We use $\text{name}(G)$, $\text{members}(G)$, $\text{capabilities}(G)$, $\text{status}(G)$, and $\text{history}(G)$ to denote the five attributes of resource group $G$.

Each member resource $R \in \text{members}(G)$ is also modeled as an object with six attributes: name, address, manager, capabilities, status, and history.

Each member resource has a name and an address where it can be accessed. We assume that each resource is managed by exactly one resource manager. Similarly, we use $\text{name}(R)$, $\text{address}(R)$, $\text{manager}(R)$, $\text{capabilities}(R)$, $\text{status}(R)$, and $\text{history}(R)$ to denote attributes of $R$.

Resource capabilities

A resource is characterized by a set of capabilities. For example, a printer might have the following capabilities: postscript, color, and duplex.

The capabilities of a resource group is a function of capabilities of its member resources:

$$\text{capabilities}(G) = \bigcap_{R \in \text{members}(G)} \text{capabilities}(R).$$

We also model a process role as an object with two attributes: name and capabilities. The attribute capabilities of a process role defines the capabilities required for a resource to assume the role. For example, a picture printing role may require a printer with color and postscript capabilities.

We say that a resource $R \in \text{member}(G)$ can assume a process role $P$ if $\text{capabilities}(P) \subseteq \text{capabilities}(R)$. Similarly, a resource group $G$ can assume a process role $P$ if $\text{capabilities}(P) \subseteq \text{capabilities}(G)$. By definition, a resource $R$ can assume a role $P$ if $R \in \text{members}(G)$ and the resource group $G$ can assume $P$. Figure 2 shows the relationship among resources, resource groups and process roles.

Resource status

There are two aspects of resource status: state and load. The former indicates whether the resource is available for work items, while the latter gives a hint of possible waiting time. We use $\text{state}(R)$ and $\text{load}(R)$ to denote the current state and load of $R$.
Each resource can be in one of the following two states: AVAILABLE and NOTAVAILABLE. For example, a printer is AVAILABLE if it is up and running and also connected to the network. A person is NOTAVAILABLE if he (or she) is on vacation.

The state of a resource group is defined to be the number of member resources that are in AVAILABLE state. Therefore, the state of a resource group of 10 member resources can be an integer between 0 and 10. 

\[
\text{state}(G) = |\mathcal{R}|,
\]

where \( \mathcal{R} = \{ R \in \text{members}(G) \mid \text{state}(R) = \text{AVAILABLE} \} \).

The current load of a resource is the amount of time the resource needs to finish all work items it has already been assigned to. In real workflow environments, the actual resource load is usually difficult to obtain. It is often estimated by, for example, the number of unfinished work items that have assigned to a resource.

The current load of a resource group is defined to be the average of its member resource loads:

\[
\text{load}(G) = \left( \sum_{R \in \text{members}(G)} \text{load}(R) \right) / |G|.
\]

### 2.2 Consistency Specification

This subsection describes specification of consistency predicate \( C \) for a resource group \( G \). As we have seen, the status of a resource group is a function of member resource status. It may need to be updated when the status of member resources have changed. Consistency predicates specify when such an update is needed.

We distinguish between two kinds of resource status changes: predictable changes and unpredictable changes. Predictable status changes are known to the GRM without notification from LRMs. While unpredictable changes are ad hoc changes that the GRM will not know unless informed by the LRMs that manage the resources.

#### Predictable status changes

Predictable resource status changes are those that always happen at specified time (often periodically). Predictable status changes usually happen to a group of resources. For example, engineers will not be available on weekends.

Consistency predicate for predictable status changes consists of three parts: resource specification, state specification and temporal specification. The resource specification is a list of resources (in the resource group \( G \)) that are affected by the change. The state specification is simply the new state (AVAILABLE or NOT AVAILABLE) of the resources after the change.

The temporal specification includes the begin time, the end time, and a specification of repeatability. The begin/end time specification consists of six fields: year, month, day, hour, minute, and second. The third part of the specification consists of three lists: days of the week list, days of the month list, and days of the year list. They indicate whether and how the change will occur periodically (every week, every month, and every year, respectively) after the begin time. Empty lists mean that the change will only occur once.

For example, the following temporal specification says that the change will occur on every Monday and Friday of each week, and every 1st and 15th days of each month in 1996.

\[
\begin{align*}
\text{Begin:} & \quad \text{year}(1996)\text{month}(1)\text{day}(1) \\
& \quad \text{hour}(0)\text{minute}(0)\text{second}(0) \\
\text{End:} & \quad \text{year}(1997)\text{month}(1)\text{day}(1) \\
& \quad \text{hour}(0)\text{minute}(0)\text{second}(0) \\
\text{Repeat:} & \quad \text{weekly\{Monday, Friday\}}, \\
& \quad \text{monthly\{1, 15\}}, \text{yearly\{\}}
\end{align*}
\]

#### Unpredictable status changes

Unpredictable status changes can happen anytime and are usually due to unpredictable situations such as a person is sick or has already been assigned with work items (possibly by other WFMSs). For example, a printer may become NOTAVAILABLE if it has broken down.

Consistency predicate for unpredictable status changes consists of an attribute name, a dependency specification and a synchronization specification. The attribute name is either state or load. Dependency between group and member resource status is as defined in Subsection 2.1.

The third part is a list of possible values for resource group attribute. It also defines interesting thresholds for member resource attribute. The attribute value of the resource group will be updated whenever these thresholds have been crossed due to changes of member resource attributes.

For example, resource group state is defined to be the number of member resources that are available. Suppose that the resource group consists of 10 member resources, the synchronization specification of \( \{2, 5\} \) means that the GRM is only interested in three cases: at least 5 member resources are available, at least 2 member resources are available, and at most 1 member resource is available. Thus, the resource group state will be updated when, for example, the number of available member resources changed from 2 to 1, or from 5 to 4. Resource group state will not be updated when the number of available member resources changed from 4 to 3.
3 Resource Management

This section describes the last component of a resource group specification, the status update procedures. We consider two general approaches: using database transactions, and using workflow processes.

3.1 Resource Management Using Database Triggers

It is very likely that resource managers (both local and global) use databases to store resource information. It is therefore natural to make use of database facilities to propagate changes of individual resource status to resource groups. The following Oracle trigger [4] example updates states of resource groups when member resource states have changed.

```sql
create trigger state-update
after update of state on resource
referencing old as old_res for each row
when (resource.state != old_res.state)
declare
 x number;
y number;
begin
 select state into x from group
 where ID = GROUP;
 select last_state into y from group
 where ID = GROUP;

 if (resource.state = AVAILABLE)
 then /* update state for GROUP */
 update group set state = state + 1
 where ID = GROUP;

 /* forward if THRESHOLD crossed */
 if (x+1=THRESHOLD) & (x-y>MAX_ERR)
 then
 update group
 set last_state = THRESHOLD
 where ID = GROUP;
 update group@grp1
 set group.state = THRESHOLD
 where group.ID = GROUP;
 update group@grp2
 set group.state = THRESHOLD
 where group.ID = GROUP;
 ...
 /* for other sites */
 endif
 ...
 /* for other thresholds */
 /* for other groups */
 else /* NOT_AVAILABLE */
 update group set state = state - 1
 where ID = GROUP;

 if (x==THRESHOLD) & (y-x>=MAX_ERR)
 then
 update group
 set last_state = THRESHOLD
 where ID = GROUP;
 update group@grp1
 set group.state = THRESHOLD
 where group.ID = GROUP;
 update group@grp2
 set group.state = THRESHOLD
 where group.ID = GROUP;
 ...
 endif;
end;
```

We assume that each resource may participate in multiple resource groups and each resource group may be used by multiple GRMs. Two Oracle tables are involved: resource for individual resources and group for resource groups. Whenever a resource changed its state, the trigger will update states of resource groups of whom the resource is a member. The trigger also propagates the group state change to all GRMs that use the group when the thresholds have been crossed.

An important assumption we have made about the proposed approach is that resource group state is relatively stable. The assumption is generally reasonable in many workflow environments. The assumption may not hold if thresholds are chosen improperly.

To illustrate this, consider a resource group of 10 member resources. Suppose that 2 and 4 are two thresholds. Now consider the situation shown in Figure 3.

![Figure 3: Resource state update example.](image)

If we propagate changes everytime a threshold is crossed, six events will be generated (resulting six remote updates), at t0, t1, t2, t5, t6 and t7. For example, the remote update at t0 is needed because the threshold 4 has been crossed (at least 4 are available). Similarly, remote update at t1 is needed as the same threshold has been crossed (less then 4 are available).

The situation is clearly undesirable and can be avoided in following ways. The first approach is to
choose thresholds properly to avoid the values of common resource group state (e.g., 4 in the example). The second approach is to allow certain errors of resource group state at remote sites. Suppose, for this particular example, that we allow the group state to be off by 1, most remote updates could be avoided. In fact, only two remote updates (at $t_0$ and $t_4$) are needed.

In the above Oracle trigger implementation, we keep the last propagated state and use it to calculate the state change since the last time the group state was forwarded to remote GRMs. Remote updates are necessary only if (1) a threshold is crossed, and (2) the difference between the current and the last propagated changes are big enough ($i$, MAX.ERR).

3.2 Resource Management Using Workflow Events

In the previous example, two pieces of information have been embedded into the trigger implementation: group definitions (GROUP, THRESHOLD and MAX.ERR) and GRM references (grm1 and grm2). Group definition is used to determine if the state change should be propagated to remote GRMs, and GRM references tell the trigger where to forward the change.

The fact that the trigger has to know in advance all of the GRMs that are interested in a resource group is undesirable. This implies that every time a GRM adds or drops a resource group, the corresponding trigger has to be modified to reflect the change. The problem can be alleviated by storing group definitions and GRM references into the underlying database, but the trigger still needs to maintain connections to all GRMs in order to do remote updates.

Another (more flexible and systematic) way to address the problem is to use workflow events. Instead of updating group states at remote sites directly, the trigger generates a workflow event which will be listened to by all WFMSs. There is a resource management process at each WFMS which updates resource groups managed by the associated GRM (see Section 4 for an example resource management process).

Comparing to database triggers, workflow events have the following advantages. First, workflow events allow more flexible resource management. There is an event associated with each resource group. To add a resource group to a GRM, the WFMS only needs to register the interest in the associated event. Events will be automatically forwarded to interested WFMSs by the workflow infrastructure. There is no change needed at local site (e.g., to the Oracle trigger implementation). Also, the triggers (or other database applications) do not need to maintain multiple connections to remote GRMs. It always sends events to the local WFMS.

Second, complicated resource management policies can be easily implemented using resource management processes. Most workflow systems (e.g., HP OpenPM, see next section) provide powerful decision making facilities (e.g., build-in rule engines). It is thus possible for different resource managers to have different policies on different resources. For example, it is very easy to program resource management processes to delay propagation of some (less important) resource changes, or propagate changes of some resources only when certain conditions (e.g., temporal) are met.

Third, workflow events also allow WFMSs to do things such as resource controlled process execution which are impossible if using database triggers only. Resource controlled process execution starts a process or an activity only when certain critical resources are available. This is possible by implementing a workflow process that monitors resource status. The resource monitoring process will listen to events for the critical resources and fire processes/activities as soon as the resources become available.

4 A System for Distributed Resource Management

In the section, we will describe an implementation of the described distributed resource management using workflow events in the context of HP OpenPM workflow management system.

4.1 OpenPM Overview

OpenPM is an open, enterprise-capable, object-oriented business process management research prototype developed by Hewlett-Packard Labs. [2]. It is a middleware service that represents a substantial evolution from traditional workflow technologies. OpenPM provides all the process enactment capabilities of workflow, but also allows evolution and modification of currently running processes to better meet changing business needs while remaining independent of changing business technology.

An OpenPM process is a directed graph comprising a set of nodes connected by arcs. There are two kinds of nodes: work nodes and rule nodes. A work node defines work to be performed by external resources, while a rule node is used to define a process flow that is more complex than a simple sequence, such as concurrent process execution and synchronization of tasks. Process flows can also be controlled via events, which are raised and subscribed by rule nodes. In OpenPM, a rule node contains a list of condition-action rules, where the condition is a Boolean expression of values such as the execution status of previous nodes, the time
at which each inward arc was fired, and other data used in the process instance, while the action fires outgoing arcs. An OpenPM rule node is executed either whenever any of its inward arcs has been traversed, or any event for which it registered has occurred.

An OpenPM system consists of an OpenPM engine and a set of business objects, resource managers, process designers, process controllers, status monitors, and event handler (Figure 4).

An OpenPM process is defined using a process designer. Business objects are external resources that perform process activities. A process, after being defined, can be started, stopped, or intervened using a process controller. A status monitor keeps status information about each individual process and loads information about the entire system.

The OpenPM engine coordinates overall business process execution. It functions as a collection of highly reliable and log-based state machines. An important task of OpenPM engine is to create execution instances of a process and step through nodes as defined by the process graph. The OpenPM engine first asks resource managers to map role specification of each work nodes to business objects which are then invoked to perform tasks specified in work nodes. An OpenPM engine is also associated with a number of event handlers which map external events (e.g., Oracle events) to OpenPM internal events and forward them to workflow processes that have registered for the events.

4.2 Resource Management in Federated OpenPM

Figure 5 shows the architecture of a federated OpenPM system (from the point of view of resource management and assignment).

Each OpenPM engine in a federated OpenPM is associated with a number of resource managers and a number of event handlers. A resource manager acts like a GRM for local resources and like an LRM for remote resources. More specifically, resource managers manage (i.e., keep track of status change) business objects at local sites and also have information about business object groups at remote sites managed by other resource managers.

Updating Resource Status

Resource managers maintain status information about only those remote business object groups that are needed for process execution by the OpenPM engine. Resource managers register to remote resource managers their interests in business object groups. They also keep, for each business object group they manage, a list of remote resource managers that have registered interested in the group.

There is a special resource management process running at each OpenPM engine whose purpose is to monitor local and remote business object groups known to the associated resource managers. Figure 6 is a graph representation of the process. The process consists of a rule node and a work node 2. The rule node listens for events corresponding to each business objects groups. The rule node fires the work node when such an event occurs. The work node either forwards the event to remote resource manages if it corresponds to a local business object group, or updates status data in the associated resource manager if it corresponds to a remote business object group.

The following are the general steps required in updating status of a business object groups.

2The actual process may be more complicated than this, in order not to miss forthcoming events when processing current events."
- A member business object status has changed. Assume, as a consequence, the group status threshold has been crossed.

- The database trigger is invoked and generates a database event.

- The database event is captured by the corresponding event handler which maps it into an OpenPM event and forward it to the resource management process.

- The resource management process invokes the business object that forwards the event to all interested remote OpenPM systems. It access local resource database for a list of remote resource managers that are interested in the business object group.

- Event handlers at remote sites get the event and forward it to the corresponding resource management processes.

- The remote resource management processes invoke corresponding business objects that update the group status data in the associated resource databases.

Resource Assignment

A key component for resource assignment in OpenPM is a resource proxy which is a middleware between OpenPM engine and business objects. A resource proxy is responsible for the second step of the proposed distributed resource management. The complete resource assignment procedure is as follows.

Step 1. OpenPM engine sends a request to a local resource manager which maps a process role (for a process activity) to either a business object or a business object group. The resource manager sends back to the OpenPM engine the address of the selected business object or the address of resource proxy for the selected business object group. If it is an individual business object, OpenPM engine invokes it directly to perform the task. Otherwise, the OpenPM engine sends a message to the resource proxy with business object group specification.

Step 2. The resource proxy first contacts the resource manager (that manages the business object group) to map the business object group to an individual business object. It then invokes the selected business object. After completion, the business object replies directly to the OpenPM engine.

5 Summary and Conclusions

In the paper, we discussed some of the problems related to distributed resource management and assignment in workflow systems. We proposed a two-step resource assignment approach which significantly reduces the overhead required to manage remote resources (i.e., to keep track of resource status). We discussed both specification and implementation issues in two-step resource management and also described a possible implementation in the context of HP OpenPM workflow system.

The general interdependent data management techniques does not address workflow resource management problem satisfactorily, as the number of resources to be managed are large and the status change of a remote resource has to be propagated immediately to the workflow system. We address the problem by managing remote resources at group level. Resource groups allow the workflow system to do resource assignment without status information of individual resources, thus significantly reduce resource management overhead.

We believe that efficient resource management and assignment is important to workflow process execution. We also realize that correct and optimal resource assignment is not always possible, due to dynamic behavior of workflow resources. The proposed approach works well only for workflow environments where most workflow tasks can be performed by multiple resources and resource group status are stable (comparing to that of individual resources). Further studies are needed for other workflow environments where the above assumptions may not be true.

Due to the space limitation, we only addressed the resource management problem in a limited scope. Many issues and extensions have been left uncovered. For example, we only discussed two-step resource assignment approach. It can easily be extended to n-step resource assignment. Also, we only considered availability and load of resource status. There are other aspects of resources that may also be useful in resource assignment.

Another point worths mentioning is the equilibrium assumption of resources in a group. In practical, resources in a group are not the same as some may be more important or capable than others. For example, some printers can print faster than others. We need to take this into account when specifying resource groups, consistency predicates and thresholds.

References


