PROCEEDINGS OF SCIENCE

Data Transfer Improvement for Temporal User Account Resources

Kazushige Saga[1]
Fujitsu Limited.
4-1-1 Kamikodanaka Nakahara, Kawasaki, Japan
E-mail: saga.kazushige@jp.fujitsu.com

Kento Aida, Eisaku Sakane, and Kazutaka Motoyama
National Institute of Informatics
2-1-1 Hitotsubashi Chiyoda, Tokyo, Japan
E-mail: {aida, sakane, motoyama}@nii.ac.jp

The computing power is still increasing year by year. The sizes of data that are generated by the computer applications are also dramatically increasing. However the network bandwidths among the computing sites are not so improved. Therefore, the number of transfer of large files between sites should be decreased. This is important especially for the distributed computing environments like grids and clouds. Many scientific jobs are workflow jobs, which consist of pre-process, main-process, and post-process. In the distributed environment, it is not unusual that these processes are run as individual jobs on different computing resources in different sites. A remarkable point of this workflow job is that it has dependency. The main and post jobs have to wait until the end of previous job. In many distributed computing environments, the computing resources use temporal user account, which is assigned for each job. Because the lifetime of this temporal user account is during the job, the job has to transfer the result files to outside storages. However, during the job running, the next job is not running because of dependency. Therefore, the working directory of the next job cannot be used as that storage, and the temporal copies of result files have to be stored in an independent intermediate storage between resources. This means that the file transfer between computing resources is done with the file transfer of the same file twice. This indirect data transfer causes prolongation of the job execution time, decline of the resources use efficiency, and network congestion. To reduce the number of file transfer, we have proposed an extended BES and JSDL for direct file transfer. However, the proposal was not discussed well for the file transfer between different type resources. In this paper, we discuss the file transfer between combinations of sixteen resource types, and propose a solution for one of issues of them.

The International Symposium on Grids and Clouds (ISGC) 2013
March 17-22, 2013
Academia Sinica, Taipei, Taiwan

1 Speaker

PoS(ISGC 2013)014

PROCEEDINGS OF SCIENCE

1. Introduction

The computing power is still increasing year by year. The sizes of data that are generated by the computer applications, e.g. e-Science applications, are also dramatically increasing. However the network bandwidth among computing sites are not so improved. Therefore, many users on grids and clouds need to make much effort for decreasing the size of files to transfer on the internet.

Account management on resources in grid infrastructure is an important issue. Currently, two types of user account management methods, the static account management method and the temporal account management method, are used in resource providers. The static account management method is the conventional user account management method, where an individual persistent account is assigned to each user who submits a job to resources. The job submitted by the user runs with the privilege of the user's static account. In the temporal account management method, the account management system manages temporal accounts in the account pool. When the user submitted a job, it assigns the user one of temporal accounts in the account pool. The lifetime of the temporal account assigned to the user is same with the running duration of the user's job; that is, the temporal account is released from the user and stored again in the account pool when the user's job finishes. Then, the temporal account in the account pool would be re-assigned to another user who submits another job. The user account is also used as the owner of the working directory. Therefore, the lifetime of the working directory is the same with the lifetime of the user's temporal account in the temporal account management method, while the lifetime of the working directory is persistent in the static account management method.

Both the account management methods have different hallmarks, and some organizations (resource providers) choose the static account management method and the others choose the temporal one. The mixture of the account management methods in grid infrastructure sometime affects the performance of workflow applications. A workflow application consists of multiple jobs with the constraints of execution order and the dependency via input/output files. When we run jobs in a workflow application on resources in different organizations, we need to transfer input/output files between the organizations. For example, let us assume the workflow application, which consists of dependent three jobs, the pre-process job, the main-process job and the post-process job. When we run the pre-process job in the organization A and the other jobs in the organization B, we need to transfer the output file of the pre-process job from the resource in the organization A to that in the organization B. The performance problem happens if the organizations use the temporal account management method. Some grid middleware cannot directory transfer the file from the resource in the organization A to that in B, and it needs to transfer the file via an intermediate storage. In other words, the middleware first transfers the file from the resource in the organization A to the intermediate storage, and it transfers the file from the intermediate storage to the resource in the organization B.

PoS(ISGC 2013)014

This paper discusses the grid middleware architecture to solve the performance problem in file transfer between organizations operated by the temporal account management method. First, we present the grid middleware architecture to directory transfer files between organizations operated with the temporal account management. The presented architecture is designed by extending the Open Grid Forum (OGF) standards, the Basic Execution Service (BES) [1] and the schema of the Job Submission Language (JSDL) [2]. In the production-ready operation of grid infrastructure, it is not realistic to assume all organizations use middleware based on our middleware architecture with the extended BES. Therefore, we can assume that organizations with different operation methods, e.g. the static/temporal account management and the original/extended BES, co-exist in grid infrastructure. Next, we categorize the combination of resource operation types based on the operation methods and discuss how we can directly transfer files in the combinations of resources. In this paper, for convenience, the resource using the temporal account management method is denoted as "temporal account resource", and the resource using the static account management method is denoted as "static account resource".

The rest of this paper is organized as follows: Section 2 presents the performance problem of file transfer between resources operated by the temporal account management method and presents grid middleware architecture to solve the problem. Section 3 discusses the file transfer mechanism in all combinations of resource operation types, and Section 4 discusses the solution to the resource availability problem. Finally Section 5 summarizes the discussion in this paper and outlines the future work.

2. Direct file transfer between temporal user account resources

In general, scientific jobs are workflow jobs that consist of pre-process jobs, main-process jobs, and post-process jobs. These jobs have dependency and run sequentially. We have focused file transfer between two jobs of them in our previous paper. Because the latter job would does different actions according to result of the former job, the result files of former job are transferred to the latter job's resource after the former job is finished. This means that the lifetime of the former job and the latter job are not overlapped. If the static account resources are used for the both jobs, because of permanency of the user account and the working directory, the files can be transferred between the resources directly even if the former job was finished. On the other hand, if the both resources are the temporal account resources, the direct file transfer between the resources cannot be done at after the former job is finished, because of the lifetime of the user account and the working directory of the jobs. In this case, the former job makes temporal copies of the result files to an intermediate storage with static account before the job is finished, and then the latter job reads the copies from the storage. Thus, the file transfer between the temporal account management resources transfers the same file two times. Figure 1 shows this file transfer flow.

As mentioned above, reducing the number of file transfer is important today. Therefore we have proposed an extended BES and an extended schema of the JSDL of OGF that can transfer

PoS(ISGC 2013)014

files between temporal account resources directly [3]. In that paper, we discussed about the file transfer between different account management type resources also, but it was not so enough. In this paper we focus the file transfer between any account management type resources.

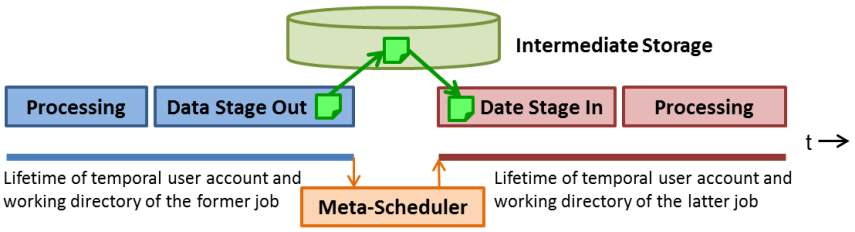

Figure 1. The conventional file transfer method between temporal account resources

The reason that cannot transfer a file directly between the temporal account resources is that the lifetimes of user account and working directory of both jobs are not overlapped. To make overlap of lifetimes of them, we proposed an extended BES that has "HOLD" sub-state and a state transition mechanism. This idea was inspired by the requirement discussion of Production Grid Infrastructure (PGI) of the OGF [4]. When the processing of former job finished, the former job is transferred into hold state and the latter job is submitted. Then the latter job transfers the file from the former job resource to the latter job resource directly. In this section the details of this mechanism and file transfer flow are explained.

Figure 2 shows the original job state model of BES and Figure 3 shows it of the extended BES. The “Running” state of the original BES is split into “DataStageIn”, “Processing”, and “DataStageOut”, and “HOLD” sub-states are added to these states as “DataStageIn:HOLD”, “Processing:HOLD”, and “DataStageOut:HOLD” respectively. And the finalizing actions of “Finished”, “Canceled”, and “Failed” states are separated as “Finalizing state" from these states. The “Finalizing” state also has the “Finalizing:HOLD” sub-state. The extended JSDL schema can describe the main state, which would be transferred to the “HOLD” sub-state. Figure 4 shows extended descriptions of JSDL. In this extended JSDL, “NOTIFY” is also defined. When the job entered the state, which is specified in the “NOTIFY” element, the resource notifies it to the job submission client. Figure 5 shows an extended BES operation for releasing “HOLD” sub-states. This operation can transfers the specified job into the specified state, if the specified state is acceptable for the current state.

PoS(ISGC 2013)014

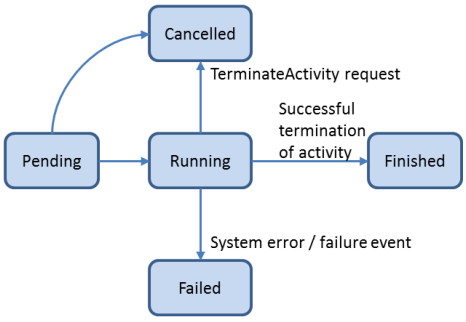

Figure 2. Job state model of original BES

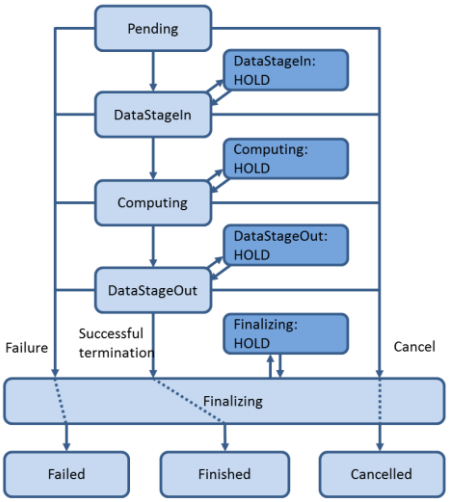

Figure 3. Job state model of proposed extended BES

```
<JobDefinition xmlns="http://schemas.ggf.org/jsdl/2005/11/jsdl">
 <JobDescription>
  <JobIdentification>
   ...
  </JobIdentification>
  <Application>
   ...
  </Application>
  <Resources>
   ...
  </Resources>
  <DataStaging>
   ...
  </DataStaging>
  <jsdl-pgi:JobFlow:xmlns:jsdl-pgi="http://www.naregi.org/jsdl/2010/04/jsdl-pgi">
   <jsdl-pgi:HoldPoints>
    <jsdl-pgi:ActivityState>Pending</jsdl-pgi:ActivityState>
    <jsdl-pgi:ActivityState>DataStageIn</jsdl-pgi:ActivityState>
    <jsdl-pgi:ActivityState>Computing</jsdl-pgi:ActivityState>
    <jsdl-pgi:ActivityState>DataStageOut</jsdl-pgi:ActivityState>
    <jsdl-pgi:ActivityState>Finalizing</jsdl-pgi:ActivityState>
   </jsdl-pgi:HoldPoints>
  </jsdl-pgi:HoldPoints>
  <jsdl-pgi:NotificationPoints>
    <jsdl-pgi:ActivityState>Pending</jsdl-pgi:ActivityState>
    <jsdl-pgi:ActivityState>DataStageIn</jsdl-pgi:ActivityState>
    <jsdl-pgi:ActivityState>Computing</jsdl-pgi:ActivityState>
    <jsdl-pgi:ActivityState>DataStageOut</jsdl-pgi:ActivityState>
    <jsdl-pgi:ActivityState>Finalizing</jsdl-pgi:ActivityState>
   </jsdl-pgi:NotificationPoints>
  </jsdl-pgi:JobFlow>
 </JobDescription>
</JobDefinition>
```

Figure 4. Example description of extended JSDL

ChangeActivityStatus
- Arguments
  - JobID to be transition
  - State transition to
- Description
  - Specified job which is described in the first argument is transitioned to the specified state which is described in the second argument
- Error
  - Authentication error
  - No such Job
  - Invalid state
  - Known state

Figure 5. Proposed extended BES operation

Figure 6 shows the file transfer flow of the proposed method. The former job is submitted with the “Finalizing:HOLD” option. When the former job entered the “Finalizing” state, the BES transfers the job into the “Finalizing:HOLD” sub-state, then notifies the transition and the end status of processing to the meta-scheduler. When the meta-scheduler received the notification, it submits the latter job with the “Processing:NOFITY” option. The latter job transfers the file as the "Data Stage In" action from the former job resource to the latter job

PoS(ISGC 2013)014

resource directly. When the file transfer finished and the latter job was moved into the "Processing" state, the latter job resource notifies it to the meta-scheduler. Then the meta-scheduler releases the "Finalizing:HOLD" sub-state with the extended BES operation.

Thus, the computing resources, which support the proposed method, can transfer files directly between resources, even if the resources use the temporal account management method.

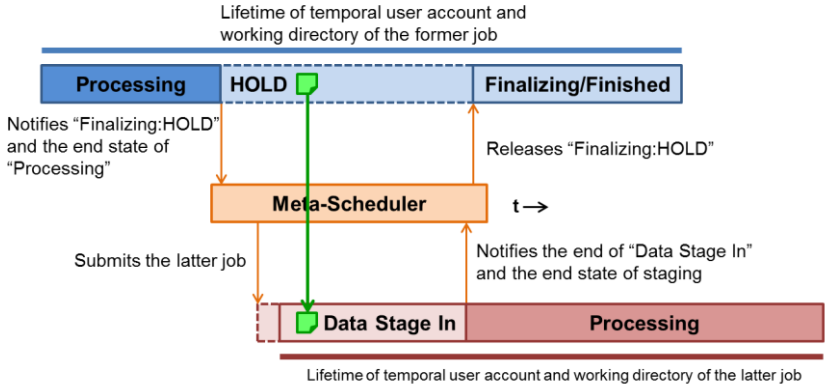

Figure 6. File transfer flow of the proposed method

3.Subsubsection

By the combination of two types of BES, original and extended, and two types of account management methods, static and temporal, there are four resource types. Therefore, for a file transfer, there are sixteen resource type combinations. In this section, we discuss about the file transfer flows of the all combination of resource types. In this study, we classified file transfer flows in six types. Table 1 shows the transfer flow types of each combination. As a result of this study, the direct file transfer can be done in all resource combinations except between resources, which use the original BES, and temporal account management. However the colored combinations in the table have a problem. In this section, the details of these file flows and the problems are discussed.

Table 1. File transfer flow types

Former job \ Latter job		Original BES		Extended BES	
		Temporal	Static	Temporal	Static
Original BES	Temporal	Indirect	Direct: Type-4	Direct: Type-5	Direct: Type-4
	Static	Direct: Type-3	Direct: Type-3	Direct: Type-3	Direct: Type-3
Extended BES	Temporal	Direct: Type-2	Direct: Type-2	Direct: Type-1	Direct: Type-1
	Static	Direct: Type-3	Direct: Type-3	Direct: Type-3	Direct: Type-3

In this study, we assumed the following resource and environmental conditions.

- The job submissions for both resources are done by a meta-scheduler, which supports the extended BES interface.

PoS(ISGC 2013)014

- Each extended BES resource has a large storage, which has enough capacity for working directories for multiple jobs. The large size files of the jobs will be stored in these directories.
- The working directory of each resource is accessible from other resources.
- The result files in the static working directory can be removed with an option of DataStaging element of JSDL, and the user of the job can specify this option. Therefore the cleanup of the files in the working directory is out of scope of this paper.

The considerable points for deciding the file transfer flows and classifying transfer types are;

- Because the resource of latter job is brokered after the former job is finished, the resource type for latter job is unknown when the former job is submitted.
- Because the file transfers of former job are described in the job document, the file transfer flows have to be decided before the former job is submitted.

These things mean that the file transfer flow has to be decided according to the resource type of former job. If there is the case, which is impossible to decide transfer flow at that time, the solution for it has to be considered.

For convenience for further discussions, the resource type is denoted as "[BES type : Account management type]", and the file transfer between resources is denoted as "→". For instance, "[Extended:Temporal]→[Original:Static]" means that the file transfer between a resource using the extended BES and the temporal account management, and a resource using the original BES and the static account management.

3.1 Indirect file transfer ([Original:Temporal]→[Original:Temporal])

As mentioned above, the making overlap of the user account and the working directory of both former and latter jobs are needed for direct file transfer. To realize this, the progress of jobs has to be controlled. Because the original BES resource has no such control method, the file transfer between the [Original:Temporal] resources cannot do the direct file transfer. This is a typical file transfer flow between conventional temporal account resources.

3.2 Direct file transfer

We classified the direct file transfer flows into the five types according to resource combinations as follows.

Type-1:

- [Extended:Temporal]→[Extended:Temporal]
- [Extended:Temporal]→[Extended:Static]

Type-2:

- [Extended:Temporal]→[Original:Temporal]
- [Extended:Temporal]→[Original:Static]

Type-3:

PoS(ISGC 2013)014

- [Extended:Static]→[Extended:Temporal]
- [Extended:Static]→[Extended:Static]
- [Extended:Static]→[Original:Temporal]
- [Extended:Static]→[Original:Static]
- [Original:Static]→[Extended:Temporal]
- [Original:Static]→[Extended:Static]
- [Original:Static]→[Original:Temporal]
- [Original:Static]→[Original:Static]

Type-4:

- [Original:Temporal]→[Extended:Static]
- [Original:Temporal]→[Original:Static]

Type-5:

- [Original:Temporal]→[Extended:Temporal]

3.2.1 Type-1 ([Extended:Temporal]→[Extended:Any])

The file transfer flow Type-1 is the flow that proposed in our previous paper and introduced in the section 2. If an [Extended:Temporal] resource is brokered as the former job resource, it is assumed that the file transfer is initiated by the latter job to use the proposed direct file transfer method. Because the latter job's resource type is unknown when the former job is submitted, the former job cannot initiate the file transfer even if the [Extended:Static] is brokered for the latter job later.

3.2.2 Type-2 ([Extended:Temporal]→[Original:Any])

As discussed in the Type-1, if the resource type of former job is [Extended:Temporal], it is assumed that the latter job initiates the file transfer. This assumption is available even if the latter job resource type is [Original:Any]. However it has a problem that because the latter job resource is an original BES resource, it has no way to notify the end of the file transfer to the job submission client. This means that the release of the “HOLD” sub-state would be postponed until the end of the latter job. This would not be big problem, because of the following reasons. The “Finalizing:HOLD” of the extended BES can be implemented that the computing nodes can be released from the job at the end of "Processing" state of the job, and can be used for another job. Therefore the resource utilization efficiency problem would not occur. And an environmental condition assumes that the storage capacity of working directories is enough for multiple jobs. Figure 7 shows this file transfer flow.

PoS(ISGC 2013)014

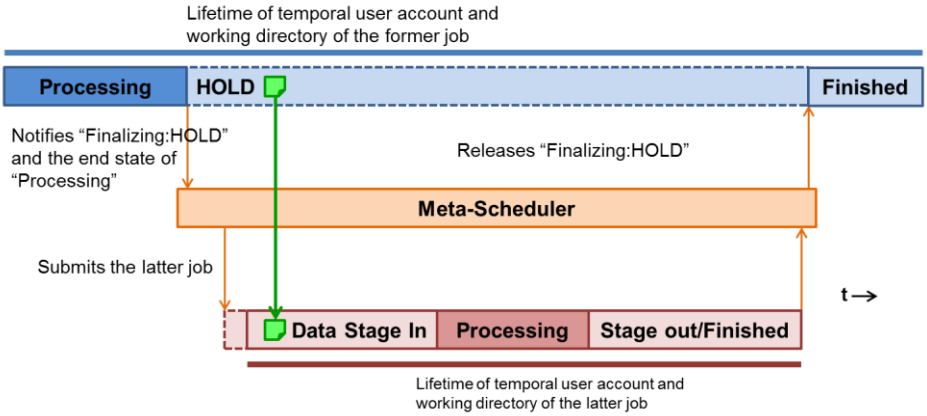

Figure 7. File transfer flow of Type-2

3.2.3 Type-3 ([Any:Static]→[Any:Any])

If the account management type of the former job is "static", the common file transfer flow for all BES types can be used. Because the result files can be left even after the former job is finished, the files are transferred by the latter job. Figure 8 shows this file transfer flow. If the latter job resource is a static account resource, the latter job can initiate the file transfer. Because the latter job resource is unknown when the former job is submitted, the former job cannot initiate the file transfer even if the type of the latter job resource is [Any:Static].

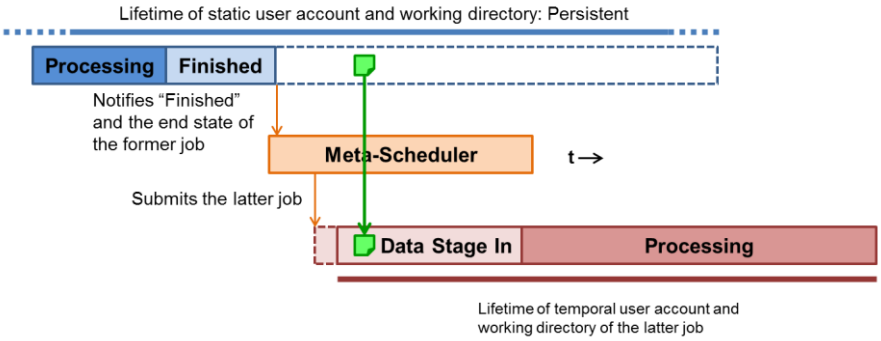

Figure 8. File transfer flow of Type-3

3.2.4 Type-4 ([Original:Temporal]→[Any:Static])

To realize direct file transfer with the [Original:Temporal] as the former job resource, because it is impossible to control the lifetime of former job, advanced brokering of the latter job resource is needed. Type-4 is the cases that an [Any:Static] resource is brokered for the latter job in advance.

PoS(ISGC 2013)014

In these cases, the former job can initiate transfers of result files to the static working directory of latter job directly. The extended BES functions are not needed. However Type-4 has a problem. In many scientific applications, it is not unusual that the jobs run long time like few days. As mentioned above, the latter job resource is brokered in advance when the former job is submitted. This delay until using the latter job resource makes many chances that the latter job resource cannot satisfy the resource requirement of latter job. For instance, the latter job resource is stopped for maintenance, is removed for replacement, and the configuration is changed. The solution for this issue will be discussed in the section 4.

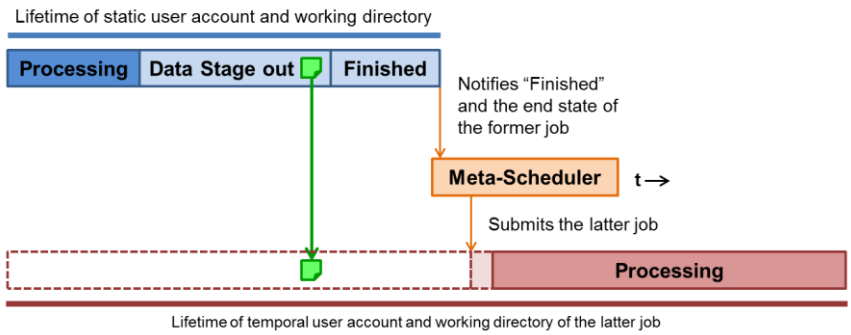

Figure 9. File transfer flow of Type-4

3.2.5 Type-5 ([Original:Temporal]→[Extended:Temporal])

As mentioned in the Type-4, if the former job resource is an [Original:Temporal] resource, the latter job resource is brokered in advance. In this combination, the latter job is submitted first with "Processing:HOLD" directive in the job document. When the meta-scheduler receives the "Processing:HOLD" notification, it submits the latter job. In the latter job, it initiates the file transfers as "Data Stage Out" of the former job. This file transfer flow has the same resource availability problem with Type-4.

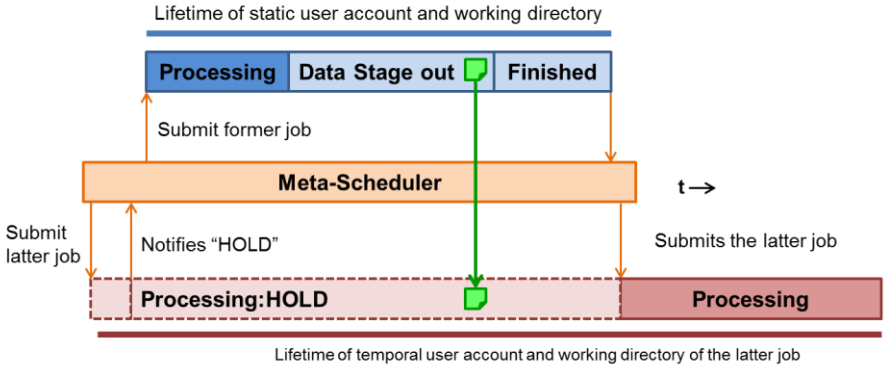

Figure 10. File transfer flow of Type-5

PoS(ISGC 2013)014

4. A solution for the resource availability problem

Because the file transfer of Type-3 and 4 is initiated by the former job and the file transfer is described in the job document of former job, the target working directory location has to be decided before the submission of former job. This is the reason for the latter job resource is brokered in advance. However the advanced brokering of the latter job resource causes the resource availability problem. One of the solutions for this issue is that a temporal working directory location is specified in the file transfer of the job document at the job submission, and is replaced with the actual location when the file transfer is initiated. To realize this, we extended the Resource Namespace Service (RNS) file catalog service [5] [6].

The RNS file catalog service can map widely distributed physical files with names of a global hierarchical namespace. To access a physical file, at first, the RNS file catalog client queries the physical location with the global name of it to the RNS file catalog server, then the client accesses the physical resource of the replied location with a conventional file access method. We modified the RNS file catalog server to solve the resource availability problem.

When an [Original:Temporal] resource is brokered, the meta-scheduler creates job-id based directory with "rns-mkdir" operation. And the "URI" of "DataStaging" element of the job document is modified. The protocol of URI is modified with "RNS", and the file path of URI is modified with a job-id based global name. Then the meta-scheduler submits the former job. At this time, the meta-scheduler doesn't broker the latter job resource. When the former job initiates the "RNS" file catalog service based file transfer, the RNS file catalog client queries the physical directory location to the RNS server with the specified job-id based name. When the RNS server receives the query with job-id based name, it requests to update the catalog of that name to the meta-scheduler. Then the meta-scheduler brokers the latter job resource of that job-id. If the brokered resource is [Any:Static], the meta-scheduler updates the physical directory location of the RNS catalog with the home directory location of the resource. If the brokered resource is [Extended:Temporal], the meta-scheduler submits the latter job with "Processing:HOLD" directive and waits the notification from the resource. And it updates the RNS catalog with the working directory location of the latter job's resource. Then the RNS file catalog server answers the physical file location to the RNS client. The RNS client initiates the file transfer of from former job's resource to that location. When the former job was finished, the meta-scheduler treats the latter job. If the latter job resource is [Any:Static], the meta-scheduler submits the latter job. If the latter job resource is [Extended:Temporal], the "Processing:HOLD" of latter job is released with an extended BES operation. Figure 11 shows this file transfer flow.

PoS(ISGC 2013)014

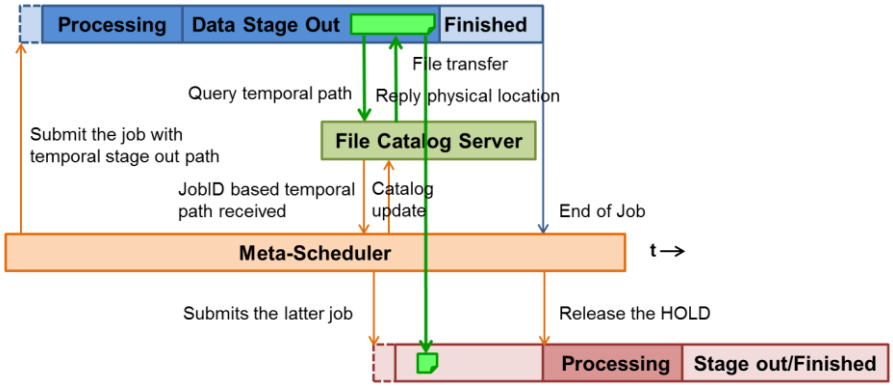

Figure 11. File transfer flow using modified RNS file catalog service

Thus, because the proposed method brokers the latter job resource when the file transfer is initiated, it can transfer files between two jobs directly without the resource availability problem.

5. Summary

We discussed a problem of file transfer of between the jobs running on temporal account resources. If the jobs have dependency, the conventional way needs a temporal copy of the transfer file in an intermediate storage between resources, because of the lifetime of user account and working directory. The former job transfers the file to intermediate storage, then the latter job reads it. This indirect file transfer causes prolongation of the job execution time, decline of the resources use efficiency, and network congestion. We introduced a solution that we proposed before. Because the solution doesn't need a temporal copy between resources, the number of file transfer can be reduced. However the previous work mainly discussed about the file transfer between the temporal account resources. There are many other type resources in grid environments. Therefore, in this paper, we studied about reducing the number of file transfer between sixteen resource type combinations. We classified the file transfer flow of these sixteen combinations into six types including the conventional indirect one and our previously proposed direct one. And we proposed the details of file transfer flows of the rest of types. Two types of the file transfer flows have a resource availability problem. To solve this problem, we proposed the extended RNS file catalog service.

We have a plan to confirm the effectiveness of the proposed methods in an interoperation environment that will be built by RENKEI [7] and some grid middleware in near future.

Acknowledgement

A part of this work is the results in the "REsources liNKage for E-scIence (RENKEI)" project. This work was supported in part by MEXT in Japan and JSPS KAKENHI Grant Number 24240006.

PoS(ISGC 2013)014

References

[1] I. Foster, A. Grimshaw, P. Lane, W. Lee, M. Morgan, S. Newhouse, S. Pickles, D. Pulsipher, C. Smith and M. Theimer, "OGSA Basic Execution Service," Open Grid Forum, 2008. [Online]. Available: http://www.ogf.org/documents/GFD.108.pdf.

[2] A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher and A. Savva, "Job Submission Description Language (JSDL) Specification," Open Grid Forum, 2008. [Online]. Available: http://www.ogf.org/documents/GFD.136.pdf.

[3] K. Saga, K. Aida and K. Miura, "Reduction of Data Transfer in Grid Computing Environments" (in Japanese), ACS39, Information Processing Society of Japan, 2012

[4] RequirementList_v7_matrix.xls, Open Grid Forum, 2010. [Online]. Available: http://redmine.ogf.org/dmsf_files/12372

[5] M. Morgan, A. Grimshaw, O. Tatebe, "RNS Specification 1.1," Open Grid Forum, 2010. [Online]. Available: http://www.ogf.org/documents/GFD.171.pdf

[6] H. Matsuda, "A Metadata Search Interface for RNS File Catalog," Open Grid Forum, 2011. [Online]. Available: http://www.ogf.org/OGF30/materials/2211/OGF31-GFS-matsuda.pdf

[7] REsources liNKage for E-scIence (RENKEI), http://www.e-sciren.org/index-e.html

PoS(ISGC 2013)014

