ISSN 1598-0170 (Print)
ISSN 2287-1136 (Online)
http://www.jksii.or.kr

DNA 서열 분석을 위한 클라우드 컴퓨팅 기반 지능형 미들웨어 설계☆

A Framework of Intelligent Middleware for DNA Sequence Analysis in Cloud Computing Environment

오 준 석[1] 이 윤 재[2] 이 봉 규[2*]
Junseok Oh Yoonjae Lee Bong Gyou Lee

요 약

차세대 유전체 해독 기술과 자동화 기술이 발전하면서 DNA 서열 분석 환경이 개선되고 있지만, 아직까지 제한된 컴퓨팅 리소스는 분석시간 단축의 장애요인으로 작용하고 있다. 대부분의 과학 워크플로우 시스템은 수 많은 기능들이 특정 시스템 환경에 맞추어 구현되어 있기 때문에 복잡하고 유동적이지 못하며, 이로 인해 기존 시스템의 컴포넌트들을 클라우드 환경의 새로운 시스템에 적용하기 어려운 한계를 지니고 있다. 본 연구에서는 대량의 DNA 데이터를 동시적으로 분석할 수 있는 가상 인스턴스 제공이 가능하며 시스템간의 상호 운용성을 개선시키기 위하여 웹 서비스, DBMS, 클라우드 컴퓨팅 기능을 지원하는 DNA 서열 분석용 미들웨어를 개발하였다. 본 연구에서 개발된 지능형 미들웨어는 DBMS를 사용하여 파이프라인 정보를 관리하고, 클라우드 환경에서 경량의 가상 인스턴스를 제공하며, 상호운용성 개선을 위하여 단순 URI와 XML을 기반으로 한 RESTful 웹서비스 기능을 제공한다.

☞ 주제어 : 지능형 미들웨어, 클라우드 환경, DNA 서열 분석, 데이터베이스 관리, RESTful 웹서비스

ABSTRACT

The development of NGS technologies, such as scientific workflows, has reduced the time required for decoding DNA sequences. Although the automated technologies change the genome sequence analysis environment, limited computing resources still pose problems for the analysis. Most scientific workflow systems are pre-built platforms and are highly complex because a lot of the functions are implemented into one system platform. It is also difficult to apply components of pre-built systems to a new system in the cloud environment. Cloud computing technologies can be applied to the systems to reduce analysis time and enable simultaneous analysis of massive DNA sequence data. Web service techniques are also introduced for improving the interoperability between DNA sequence analysis systems. The workflow-based middleware, which supports Web services, DBMS, and cloud computing, is proposed in this paper for expecting to reduceanalysis time and aiding lightweight virtual instances. It uses DBMS for managing the pipeline status and supporting the creation of lightweight virtual instances in the cloud environment. Also, the RESTful Web services with simple URI and XML contents are applied for improving the interoperability. The performance test of the system needs to be conducted by comparing results other developed DNA analysis services at the stabilization stage.

☞ Keyword: Intelligent middleware, Cloud environment, DNA sequence analysis, Database management, RESTful web services

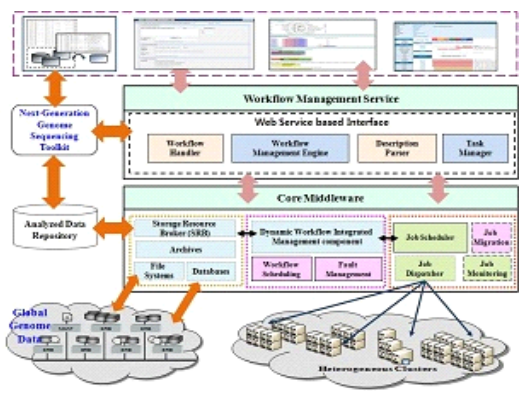
1. INTRODUCTION

The development of information technologies and computing resources has reduced the time required for analyzing DNA sequences. However, existing platforms still have a limitation on storage, computing resources, and data sharing. Many platform vendors make dedicated efforts to

1 Communications Policy Research Center, Yonsei University, Seoul, 120-749, Korea.
2 Graduate School of Information, Yonsei University, Seoul, 120-749, Korea.
* Corresponding author (bglee@yonsei.ac.kr)
[Received 31 October 2013, Reviewe 4 November 2013, Accepted 18 December 2013]
☆ This work was supported by the Industrial Strategic technology development program, 10040231, "Bioinformatics platform development for next generation bioinformation analysis" funded by the Ministry of Knowledge Economy (MKE, Korea)"
☆ A preliminary version of this paper appeared in APIC-IST 2013, Aug 12-14, Jeju Island, Korea. This version is improved considerably from the previous version by including new results and features.

adopting the next generation scientific workflow systems to overcome these limitations. As the representative scientific workflow system, the myGrid project has progressed to develop an integrated environment between E-Science and biological information in Europe. In spite of these moves, modules for scientific experimentation have encountered several problems, such as lack of interoperability, because they are invented by the needs of the individual researcher at the laboratory level without consideration for standardization or interoperability.

In this paper, we have developed workflow-based middleware as a part of a national project. The project is aimed for developing the next generation bioinformatics platform. The objectives for development are 1) a bioinformatics workflow for DNA analysis based on high-speed cluster or cloud computing, 2) Bio Applications and 3-D Visualization, and 3) DNA analysis including microbial community analysis [1]. The workflow-based middleware developed in this paper plays the role of the workflow management system based on cloud computing. It submits activity information to the scheduler in supercomputing farm and monitors the workflow status by the RESTful Web services and database interfaces.



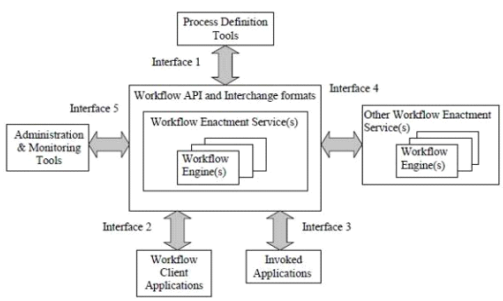
(그림 1) GiSys 플랫폼 아키텍처
(Figure 1) GiSys Platform Architecture

The rest of the paper is as follows. In Section 2, examine related research, in Section 3, introduce framework and interface of workflow-based middleware, in Section 4, describe the detailed function of workflow-based middleware, and in Section 5, discuss conclusions and future work of our research.

2. WORKFLOW STUDIES

2.1 SCIENTIFIC WORKFLOW

The workflow technology is introduced to area of business to realize OA (Office Automation). The workflow is a system that can define, manage and execute the flow of activities through software whose order of execution is driven by a computer representation of the workflow logic. Workflow system users are able to achieve the integrated environment with distributed service through the systems. The WfMC (Workflow Management Coalition), founded in 1993, is a global organization of developers, consultants, and research groups engaged in workflow and BPM (Business Process Model). WfMC created the standards for workflow and suggested XPDL (XML Process Definition Language) and Wf-XML as the representative standards, which are process design formats for storing the visual diagram [2]. It also suggests the workflow reference model shown in Figure 2 [3,4].



(그림 2) 워크플로우 참조 모형
(Figure 2) Workflow reference model

The workflow reference model has five interfaces. Interface 1 supports the exchange of data for the process definition between BPR (Business Process Re-engineering) tools, workflow systems, and process definition repositories. It helps workflow users to select an appropriate tool for the business process lifecycle. Interface 2 was developed for ap-plication integration in different workflow systems. The WMS (Workflow Management System) developers are able

to design the portable client and reuse it to other WMS. In-terface 3 provides an interface framework for integrating applications to the systems or services in other industries. Specifically, it supports the common interfaces for enterprise legacy applications. Interface 4 was developed for process automation in different environments. Interface 5 is for the administration of workflow cases across systems by the specification of a common model [4].

(표 1) 비즈니스 워크플로우와 과학 워크플로우의 비교
(Table 1) Comparison of business workflow and scientific workflow

Category	Business Workflow	Scientific Workflow
Target	Transaction Process and State	Problem Solving Knowledge
Verification & Error correction	Consistent way with mutually agreed policy	Learning from Mistakes Validation of the intermediate results is important
Reusable	Not Important	Important
Flexibility	Fixed	Based on experiments Rapid & Flexible
Data Type	Small amounts of data Structured data type	Large amounts of data Unstructured data type and diverse formats
Process Flow	Data and Control flow separately proceeds Object state is important in process flow	Data and Control flow proceeds together Data value is important in process flow.

With the varied usage of workflow systems in industry, it began to be applied in scientific areas, which require complex computations or analyses [5]. The workflow systems create the experimentation process to follow prescribed procedures. Complex experiments in scientific fields, such as bioinformatics, require users to check each step of the whole experimental process. The scientific workflow provides users the ability to monitor the process of experiments and get the final results by the automated functions in the workflow. The business workflow and the scientific workflow were invented for the same reason to automatically control the job processes, but scientific workflow has the high-tech functions to address complex computations. The business workflows progress around the process flow; conversely, the scientific workflows progress around the data flow. The scientific workflows mainly perform data management, analysis, simulation, visualization of experiment, and proof scientific demonstration [6]. Particularly, the workflow for the life sciences should have functions to handle data conversion with each experimental process, and record the relationship between the results in each step. In addition, the workflow should make a connection between the extracted results and variety of biological applications, and provide a variety of visualization tools with plug-in type [7]. Table 1 shows a comparison of the business workflow and the scientific workflow.

The Kepler project [8], as introduced on its Website, “is designed to help scientists, analysts, and computer programmers create, execute, and share models and analyses across a broad range of scientific and engineering disciplines and can operate on data stored in a variety of formats, locally and over the Internet, and is an effective environment for integrating disparate software components”. Kepler is built upon the Ptolemy II system based at the University of California at Berkeley. This is the analysis tool for scientific circles of Workflow. In Kepler, the focus is on the workflow of GUI-based and can be performed effectively in a distributed computing environment [9].

Taverna is a Grid-aware workflow management system. It is developed as a part of MyGrid project for various bioinformatics analysis based on workflow at the University of Manchester [10]. The workflow systemprovides transparency, semantically-enabled, loosely-coupled middleware to support researchers that successfully perform data-intensive experiments on distributed resources. Taverna is also composed of the Workbench engine and it uses an independent and proprietary language, which is SCUFL (Simple Conceptual Unified Flow Language) [11]. The SCUFL language allows users to define a workflow for local groups or remote services which are connected by allowing the coordination of services.

BioWBI-WEE is a workflow system for bioinformatics based on Web services, developed by IBM alphaworks, which is an enormous IT solution company [12]. BioWBI-WEE is composed of two parts. One is the BioWBI

(Bioinformatics Workflow Builder Interface) which models the workflow based on Web services. Its users are able to log into the system, create their own custom workflow, and access workflows created by other users. The second part of BioWBI-WEE is a WEE (Workflow Execution Engine) which executes a workflow made by BioWBI and returns the results to the interface. The results are collected and checked by comparing them with other data. The general workflow systems depend on application processing at a local level, but the user of BioWBI also has access to the presentation layer at a local level, that is, including application processing, access control, and data storage management by BioWBI [13].

Triana is an opensource scientific workflow system developed by the GridLab project [14]. Triana is designed to define the processes and monitor workflows which include the processes. Triana's workflow designer, called the toolkit, allows users to compose workflows graphically by dragging and dropping components into a workspace [15]. Triana provides interoperability by supporting multiple languages to use a plugged in function. Triana's multiple languages include BPEL (Business Process Execution Language), WSFL (Web Services Flow Language), Petrinet formats, and DAG (Directed Acyclic Graph) [6].

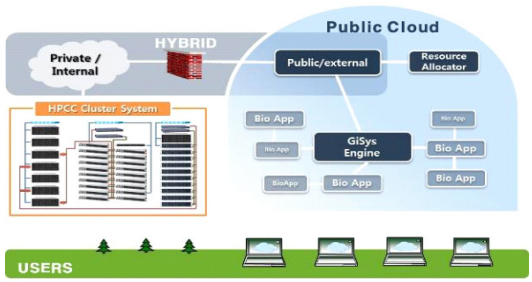
Galaxy is a software system, which provides support to experimentalists, developed by the Pennsylvania State University and Emory University [16]. Galaxy can give specific functions to experimentalists through simple interfaces to powerful tools, while automatically managing the multistep computational details. Galaxy also provides GUI (Graphical User Interface) to users with data in progress. It supports multiple formats related to bioinformatics and can upload functions in which the user can upload data through URL. Galaxy allows experimentalists without informatics or programming expertise to perform complex large-scale analysis with just a web browser.

2.2 CLOUD-BASED WORKFLOW

Recently cloud computing environment has been applied to the areas of bioinformaticsin order to overcome growth limits [17]. The cloud computing offers the benefits of integrating data and reducing the IT cost. Stein suggested that migrating to the cloud computing makes DNA sequencing cheaper [18]. Because the cloud computing base workflow systems have advantages compared to existing workflow system, it can serve as the middleware service to facilitate the usage of cloud services. The cloud computing base workflow systems can be used in many complex e-science applications, which include bioinformatics, earthquake modeling, weather forecasting, and high-level physics [19]. Nebro et al. and Schadt et al. provided the practical solution in order to achieve huge amount of time reduction in genomic analysis [20,21].

The cloud computing base workflow systems have several advantages. 1) A data interface standardizes among various organizations. 2) The different genetic analysis algorithms can be applied through encapsulation. 3) A platform can respond to storage changes in the event of a rapid increase in data and provides data integration through virtualization. GeoSpiza, which is member of the software development community of Applied Biosystems, introduced their web based bioinformatics analyzing system and is based on Amazon cloud services. By introducing cloud computing on the bioinformatics workflow system, the companies can achieve effective data integration with standard API (Application Program Interface) and save TCO (Total Cost of Ownership)through operating distributed computing and virtualization [22].

The national project is for conducting DNA sequence analysis based on NGS (Next Generation Sequence) data in Korea. The GiSYS (Genome Informatics System), which is the integrated bioinformatics analyzing platform, is developed as the main product of this project [23].



(그림 3) 클라우드기반 GiSys 플랫폼 구조
(Figure 3) Structure of cloud-based GiSys Platform

(Table 1) shows the functions of GiSYS middleware which is developed on the basis of the KT cloud services and the Clunix parallel computing environment in Korea. The GiSYS middleware consists of a workflow designer, workflow engine, and resource scheduler.

(표 1) GiSYS 플랫폼의 기능
(Table 1)The functions of GiSYS platform

Category	Description
Workflow Designer	− Providing the design tools for defining the module and the pipeline − Providing drag and drop components
Workflow Engine	− Parsing XML contents designed in Workflow Designer − Providing RESTful Web services to the user interfaces for running pipelines − Executing pipelines or modules following the workflow − Consuming RESTful Web services from Scheduler for running job commands
Scheduler	− Scheduling the analysis commands on high-speed cluster and public cloud computing environment − Providing RESTful Web services to Workflow Engine for submitting job and providing results
Web UI	− Providing user log-in function − Consuming RESTful Web services from Workflow Engine for running pipelines

The GiSYS uses high-speed cluster or public cloud computing for analyzing bioinformation. It uses lightweight virtual instances for executing modules on cloud machines in effective ways. The virtual instance includes OS (Operating System) and bioinformatics analysis applications which are BWA Tool, Samtools, Bcrtools, and Vcfutils [1].

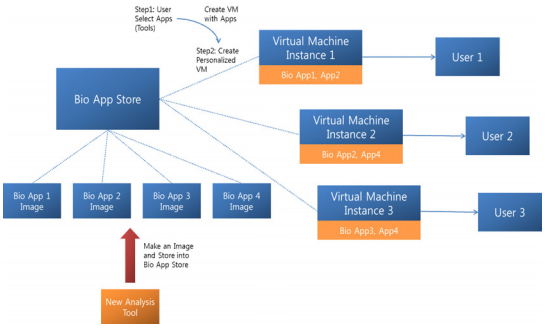
3. CLOUD-BASED PIPELINE ANALYSIS FRAMEWORK AND ALGORITHMS

3.1 CLOUD-BASED ANALYSIS SYSTEM

The DNA sequence analysis generally requires large amounts of storage and high performance computing servers due to the size of DNA sequence data. In order to overcome the limitation of computing resources on local analysis machines, distributed storage and servers are traditionally used with the development of the computer network environment. Despite the benefits from the distributed computing environment, the analysis service providers have to spend a significant amount for building analysis infrastructures. The analysts are able to save on the cost for analysis with the development of cloud computing technologies and grid network.

The DNA sequence analysis platform suggested in this paper is based on the cloud computing technologies as well. The cloud service provided is the ucloud which is the popular cloud service in Korea [23]. It is provided by KT (Korea Telecom) which is the representative telecommunication company in Korea. The ucloud which is applied on the middleware in this research provides users the desktop and mobile cloud services including storage, web service engine, data-base, and virtual middleware services. KT is currently developing the cloud computing services based on Hadoop to analyze large amounts of data in a short time, especially for DNA sequence analysis. The cloud computing infrastructures support the sequence analysis middleware by providing personal storage and virtual instance for fast and economic analysis. The analysis applications or tools are uploaded to the application data-base (Bio App Store) and managed in the application management server. The Bio App Store is the cloud based database service which is provided by ucloud and it includes the analysis utilities, such as ‘Bwa alignment’ and ‘samtools’.When an analyst log-in using their personal account, the middleware provides each analyst the virtual analysis instance having analysis tools, personal storage and WMS-based user interface. The virtual analysis instances in Figure 1 are created and run on the cloud servers in ucloud services. The analyst is easily able to create the personal analysis pipeline by the user interface.

When the analyst adds the activity on the pipeline by drag and drop, the middleware imports selected application from Bio App Store as activity to the pipeline. The applications are run in the cloud-based supercomputing farm according to the flow of pipeline. The supercomputing farm exists to run analysis utilities in a grid computing environment. The utilities include ‘alignment’, ‘variance calling’, ‘reporting’ and they are based on the existing open



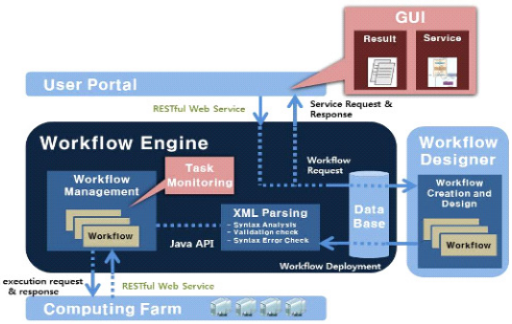
(그림 4) DNA 서열분석 미들웨어에서의 클라우드 서비스 역할
(Figure 4) Role of cloud services in DNA sequence analysis middleware

source-based DNA analysis tools. The grid center services of Clunix, which is supercomputing service provider in Korea, are implemented in this system [25]. Since the analysis tools compute DNA sequences in a grid network environment, the analysts are able to get fast and convenient analysis results regardless of the number of users and the number of running applications. Also, the user uses a web browser to connect to the middleware and does not require high performance client devices. Therefore, the middleware bring the benefits to analysts by providing real-time analysis tools in a ubiquitous environment.

3.2 WORKFLOW MIDDLEWARE FOR PIPELINE ANALYSIS

The pipeline is defined as the analysis sequences and set of modules. The module is defined as a process to execute the analysis command. The workflow-based middleware designed in this research is shown in Figure 5. It has a role to receive the requests from the user interfaces, to send job requests to the scheduler via RESTful Web services and tomonitors pipeline processes

The ‘Pipeline and Module Monitor’ storespipeline and module execution information into a database. It also retrieves pipeline process information from the database and delivers it to the user interface. Second, the ‘Web Service Controller’has a Web Service provider and consumer. The Web service provider receives request information from the user interfaces and runs workflows for analyzing the pipelines. It also returns the XML type response information about a successful run message or an error message. The Web service customer can request ‘job submission’, ‘monitor a job’, and ‘get job information’ Web services to the scheduler. It receives response XML message which include ‘job number’, ‘job progress’ and ‘job monitor information’ from the scheduler as well. The ‘Web Service Controller’ supports the GET method with URI (Uniform Resource Identifier) or the POST method with URI and XML for requestingWeb services.

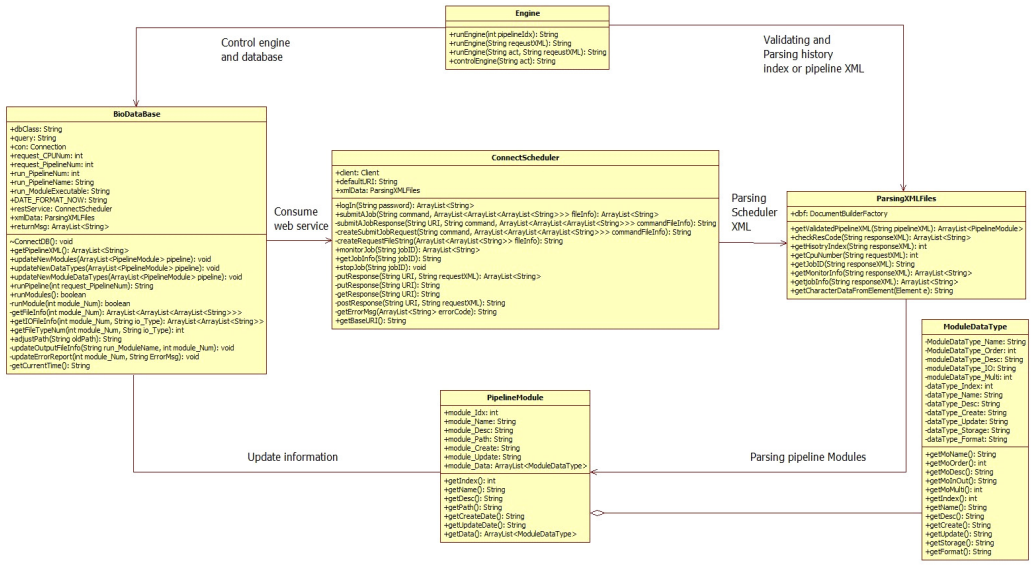


(그림 5) 클라우드 환경에서의 파이프라인 분석용 미들웨어
(Figure 5) Workflow-based pipeline analysis middleware in cloud environment

The last part of workflow-based middleware is the “Pipeline XML Validator and Parser” and it validates pipeline XML content based on a schema file, which are designed in the workflow designer. It also parses the XML file and stores new module information into the database, such as the command name, input or output file names, and module orders. This part plays the role not only to validate pipeline XML contents but also to validate request and response XML content which are used in the ‘Web Service Controller’. The middleware is implemented in Java programming language and the Jersey API is used to build the RESTful Web services. Also, The MySQL is used as a DBMS (Database Management Systems).

3.3 PROGRAM INTERFACE OF MIDDLEWARE

The workflow-based middleware has six modules and its class diagram for API is shown in Figure 6. It consists of the Engine class, the Bio-DataBase class, the ConnectScheduler



(그림 6) 워크플로우 기반 미들웨어의 클래스 다이어그램
(Figure 6) The class diagram of workflow-based middleware

class, the ParsingXMLFiles class, the Pipeline-Module class, and the ModuleDataType class. The Engine class has functions for the Web service provider while the ConnectScheduler is the class for connecting to the scheduler. The Engine class has three overloaded functions to start the middleware engine. When the user interface calls one of the functions consuming Web services,the engine class starts the engine by transferring the job id to the scheduler or it requests job status information, which is obtained from the user interface, to the scheduler. The BioDataBase classrelates to the execution of query statements for storing and retrieving pipeline status information.

This class exists in the middleware in order to manage the database. The database includes data for transaction time, update status, and file information. The transaction time is stored and managed in the database because of checking the start and end time of module running, and the order of a module in each workflow. Also, because the actual DNA sequence data and analysis result files are not stored in the engine server but stored in the scheduler server the engine has to have the reference for the information of data and results in the scheduler server. The database has the meta information of data, status, and the DNA analysis results in the scheduler. The BioDataBase classalso has functions for running pipeline and modules. The ParsingXMLFiles class includes the functions and properties for validating and parsing XML files on the basis of a schema file which is predefined the middleware. The validator and parser are implemented by Java Simple SAX (API for XML) in this class. There are two classes for storing module information and input/output file information.

The ConnectScheduler class is implemented for connecting the scheduler and consuming the Web services in the scheduler. This class receives the job id and request XML information from the user interface via Engine class. It also passes the information form user interface to the scheduler. The class also receives the information from the scheduler by consuming Web services. The Web services from the scheduler provide the job status, process steps, and monitoring results to the engine. The job is the unit in the scheduler, which is the module in the workflow definition. Since the scheduler is running in the cloud environment which is based on the parallel computing and the data is processed in different computer, the job number or job id is

important to find the status and result of unique module in the simultaneous processes.

The BioDataBase class plays a role to run pipelines and modules following the flow script. When the class runs a pipeline, it executes a query to get pipeline information from the table, then updates the pipeline start time and status information. This class calls the runModules function (See Figure 7) which retrieves the number of modules and module index, then runs each module by calling Web Service Consumer functions in the ConnectScheduler class.

If all modules in the pipeline are completely executed, theclass updates pipeline end time and status information, and returns the message to the Engine class for providing response XML to the user interface. It also plays the role of returning error information to the user interface via the Engine class when the error occurs in the scheduler as well. When the scheduler returns error information to the workflow-based middleware, the error information is stored in an error table in the database and the running process is stopped.

The Engine class has override functions to receive request information from the user interface and run pipeline in two methods. The class receives URI including the web server address with pipeline index from the user interface. The class splits the pipeline index and runs pipeline by creating a database object and calling the runPipeline function as shown in Figure 8.

```
Input: request_PipelineNum (pipeline number to run)
Output: returnString (pipeline complete message)

Procedure runPipeline(request_PipelineNum)
Start
 query <- "SELECT user_idx, pipeline_idx, name FROM History WHERE
          idx=" + request_PipelineNum;
 For each pipeline in History table
   select information and assign valeus to user_Num,
   run_PipelineNum, run_PipelinName variables
 End

 query <- "UPDATE History SET start_date='" + getCurrentTime() + "',
          status='G' WHERE idx=" + request_PipelineNum;
 Update Pipeline status information to start by
 calling getCurrentTime() function and running update query

 isError <- Call runModule() function in order to run all modules
          in each pipeline module and check the error
 If isError is true
   finish the function process and return error message
 End

 query = "UPDATE History SET end_date='" + getCurrentTime() + "',
         status='Y' WHERE idx=" + request_PipelineNum;
 Update Pipeline status information to end by
 calling getCurrentTime() function and running update query
 return pipeline complete message
End
```

(그림 7) 파이프라인 실행 알고리즘
(Figure 7) Algorithm for running a pipeline

The function returns XML type information which is the pipeline execution message or the error message. In the case of the POST method, it runs the same named function that has the request XML string for the input variable. The request XML has information for running a pipeline. This override function creates the object and runs the functions in order to validate and parse the request XML, then executes the same processes as the function for the GET method.

The BioDataBase class contains the function which runs eachmodule in a running pipeline. The function also updates the module start time and status information. The workflow-based middleware submits a job and gets a job number by RESTful Web services from the scheduler. Since 'job submission' is a POST Web service, it requires the request XML string for the input. So, the request XML is created before consuming the 'job submission'Web service. It includes the command name, the number of input data, input data names, the number of output data, and output data names as elements.

```
Input: request_PipelineNum (pipeline number to run)
Output: returnString (pipeline complete message)

Procedure runPipeline(request_PipelineNum, request_CPUNum)
Start
 query <- "SELECT user_idx, pipeline_idx, name FROM History WHERE
          idx=" + request_PipelineNum;
 For each pipeline in History table
   select information and assign valeus to user_Num,
   run_PipelineNum, run_PipelinName variables
 End

 query <- "UPDATE History SET start_date='" + getCurrentTime() + "',
          status='G' WHERE idx=" + request_PipelineNum;
 Update Pipeline status information to start by
 calling getCurrentTime() function and running update query

 isError <- Call runModule() function in order to run all modules
          in each pipeline module and check the error
 If isError is true
   finish the function process and return error message
 End

 query = "UPDATE History SET end_date='" + getCurrentTime() + "',
         status='Y' WHERE idx=" + request_PipelineNum;
 Update Pipeline status information to end by
 calling getCurrentTime() function and running update query
 return pipeline complete message
End
```

(그림 8) 사용자 인터페이스 웹서비스 제공자 알고리즘
(Figure 8) Algorithm for providing Web services to UI

After the request XML is created, the function consumes the Web services by calling the submitAJob function to run a module. The URI of the scheduler with request XML

needs to be predefined for consuming the Web service. The submitAJob function not only consumes the Web service but also validates and parses the response XML. If the parsed XML includes an error element, the function considers that the module has an error. The function updates time, module number, and the error message in the error table. It also stops running the module and returns the error status (true value) to runPipeline function in order to stop the pipeline. However, if the Web service returns the XML including job id, the submitAJob function gets a job number and it is used in the URI for monitoring a job and getting job information.

For monitoring module status, this function calls the monitorJob for consuming the GET Web service.The request of the Web service is submitted by connecting to the scheduler web server with URI while the URI string includes the job number. The response XML of the Web service has the progress elements. The progress shows the status of a job and becomes 100 when the module is successfully executed. The runModules function shown in Figure 9 periodically calls the monitorJob function to check the status of the module. The cycle is set at five seconds and the runModules function keeps parsing the response XML and checking the progress element every five seconds. It stops calling the monitorJob function and goes to the next step when the value of the element is 100.

```
Input: module_Num (module number which needs to run)
Output: errorStatus  (status of module, true: the module has an error, false: the module is successfully executed)

Procedure runModules(module_Num)
Start
  query <- "UPDATE PipelineModuleHistory SET start_date='" + getCurrentTime() + "status='G' WHERE
          history_idx='' + request_PipelineNum + " AND pipeline_module_idx='' + module_Num

  Update module status information in running pipeline to start by calling getCurrentTime() function and running
  update query

  Create job command and input/output file information XML string in each module with selected  module
  information (request XML for submit a job Web Serivce)

  returnMsg <- run submit a job RESTful Web Service by calling submitAJob() function with module path and job
  command (POST method with request XML)

  If returnMsg has an error message
    Handle error by updating ErrorReport table  and returning errorStatus (true) to runPipeline function
  Else
    jobID <- Get job Id from returnMsg

    Do
      returnMsg <- run monitor a job RESTful Web Service by calling monitorJob() function with jobID
      If returnMsg has an error message
        Handle error by the same method for the submit a job RESTful Web Service
      Else
        Request run monitor a job RESTful Web Service at every 5 seconds until the progress is 100%
      End
    End
    returnMsg <- run get job information RESTful Web Serive by calling getJobInfo() function with JobID

    Update module status information and output file information in each related table
  End
 return errorStatus (false)
End
```

(그림 9) 컴퓨팅 서버 웹서비스 소비자 알고리즘

(Figure 9) Algorithm for consuming Web services from computing scheduler server

The function gets job information by calling the getJobInfo function to retrieve job information when the module is successfully executed and updates the module end time, status, and output file information. Conversely, the function handles an error in the same manner as handling the error in the submitAJob function if the response XML of the Web services contains an error element. The XML information is validated and parsed from the ParsingXMLFiles class by creating the parser object. The class contains the functions for controlling XML strings based on Java SAX API. All functions in the class call the XML validator function which imports XML schema,creates the schema factory from SAX API, and checks grammar in the XML contents. The class has two parts for parsing XMLs after the validations. The first part includes the functions for parsing the request XML from the user interface and the response XML from the scheduler. Figure 10 shows examples of the functions to get pipeline index from the request XML and job information from the response XML.

```
Input: requestXML (Web Service request XML from the user interface)
Output: historyIdx (pipeline history index)

Procedure getHistoryIndex(requestXML)
Start
  Create XML document builder in order to parse element in the request XML from the user interface
  historyIdx <- get the element value in "historyIdx" tag in request XML
  return historyIdx
End

Input: reponseXML (Web Service response XML conveyed from the scheduler)
Output: jobInfo (the array having job information that is successfully executed)

Procedure getJobInfo(responseXML)
Start
  Create XML document builder in order to parse element in the response XML from the scheduler
  jobInfoNode <- get "JobInfoResponse" node information in the response XML
  state, progress, priority, submission, start, end, exit <- get the element value in "State", "Progress", "Priority",
        "SubmissionTime", "StartTime", "EndTime", "ExitCode" node under jobInfoNode in the response XML

  cpu <- get the element value in "CPU" node under jobInfoNode in the response XML
  maxcpu, mincpu <- get the attribute value in "maxreq", "minreq" attribute in "CPU" node

  runnable <- get the element value in "MaxRunnableTime" node under jobInfoNode in the response XML

  memory <- get the element value in "Memory" node under jobInfoNode in the response XML
  maxmem, minmem <- get the attribute value in "maxreq", "minreq" attribute in "Memory" node

  master, node <- get the element value in "MasterNode", "Node" node under jobInfoNode in the response XML
  Adds state, progress, priority, submission, start, end, exit, maxcpu, mincpu, runnable, maxmem, minmem, master,
  node on jobInfo array
  return jobInfo
End
```

(그림 10) 컴퓨팅 서버 웹서비스 파서 알고리즘

(Figure 10) Algorithm for parsing Web service XML contents from computing scheduler server

The parser is created based on Java SAX API and the function creates the XML document builder, then it gets values in requested elements. The pipeline history index is obtained by parsing the "historyIdx" element in the request XML which is delivered from the user interface. In the same

```
Input: pipelineXML (new pipeline XML which is designed in workflow designer)
Output: pipelineInfor (pipeline information which is created by parsing new pipeline)
Procedure getValidatePipelineXML(pipelineXML)
Start
  Create XML document builder in order to parse element in the request XML from the user interface
  moduleNode <- get "module" node information in the pipeline XML
  For each module
    moduleIdx <- get the element value in "mId" node under moduleNode in the pipeline XML
    moduleName <- get the attribute value in "moduleName" attribute in "mId" node

    descript, path, create, update <- get the element value in "moduleDescription", "moduleExecutable",
    "moduleCreateDate", "moduleLastupdateDate" node under moduleNode in the pipeline XML

    dataNode <- get the element value in "moduleDataType" nodeunder moduleNode in the pipeline XML
    For each data
      dataName, order, index <- get the attribute value in "moduleDataTypeName", "moduleDataTypeOrderNum",
      "datatypeIdx" attribute in dataNode

      moDesc, molo, moMulti, dataName, dataDesc, dataForm, dataCreate, dataUpdate, dataType <- get the element
      value in "moduleDataTypeDescription", "moduleDataTypeIoType", "moduleDataTypeParallelizable", "dataTypeName",
      "dataTypeDescription", "dataTypeFormatName", "dataTypeCrteateDate", "dataTypeLastupdateDate",
      "dataTypeStorageType" node under dataNode in the pipeline XML

      Create ModuleDataType object with each data information parsed from pipleline XML
      moduleData <- append object to moduleData array
    End
  Create PipelineModule object with each module information parsed from pipeline XML and moduleData array
  pipelineInfo <- append object to pipeline array
  return pipelineInfo
End
```

(그림 11) 워크플로우 엔진 파이프라인 파서 알고리즘
(Figure 11) Algorithm for parsing pipeline XML contents in workflow engine

way, the getJobInfo function parses the response XML from the scheduler to get job information that is completed.

The function also checks attributes in the response XML for getting job information. The class has the function for parsing the pipeline XML which is defined in the workflow designer and Figure 11 presents the algorithm of the function. It playsthe same role as the functions for Web service XML parser to parse the pipeline XML. In addition, the function creates two objects for storing input/output file type information and module information in the pipeline. A pipeline is designed in the workflow designer and it is stored as XML content in the database. The workflow-based middleware checks whether new pipeline XML information exists in the database when a user or an administrator requests to check a new pipeline. If the workflow-based middleware finds a new pipeline, it calls this function and parses the XML. Since the pipeline XML has different types of modules and a module has many input/output file types, the function creates a new object for storing information.

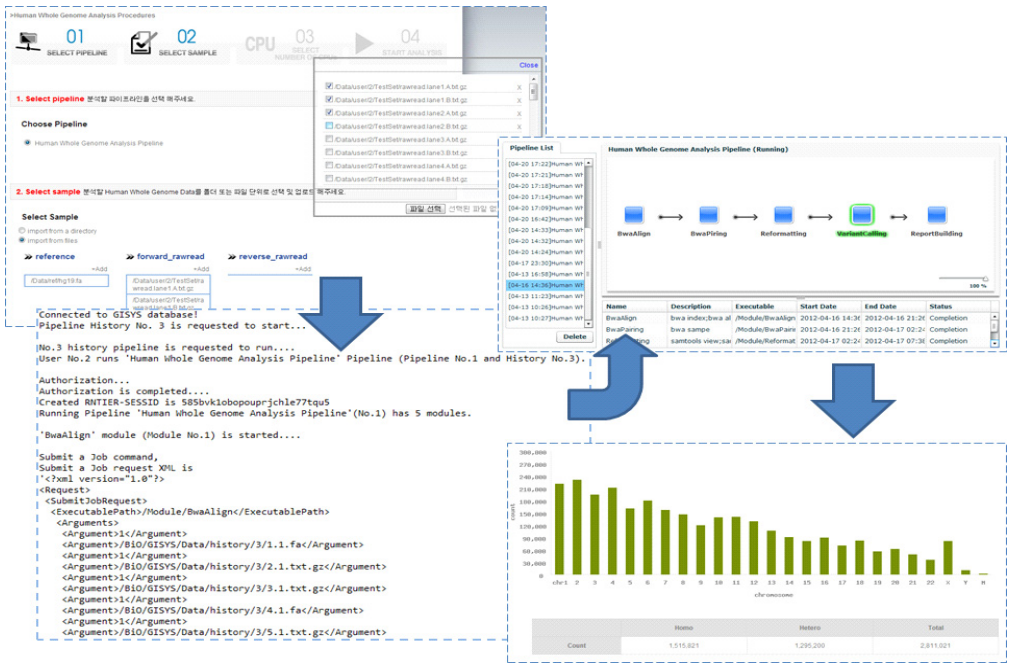
For instance, if the pipeline has two modules and each module has three file types, the function creates three objects for file type information and stores them in one array. It also creates two objects, which include the file type array, and stores them in one array. Therefore, the created pipeline array has six file types. The parsed information from the XML needs to be stored into the database when the pipeline or the module is newly created. So, the BioDataBase class has the function to retrieve the database to check new module and input/output file type. The function uses the array which is created above function in order to compare new pipeline XML information with the information in the database.

4. IMPLEMENTATION RESULTS

Figure 12 shows example results for the execution of workflow-based middleware. The example pipeline consists of five modules. The relation of analysis modules for DNA pipeline is more complicated than the one displayed in this example. For instance, the process for the BWA (Burrows-Wheeler Aligner) alignment for each DNA sequence number is compiled into one batch file command and the process for all sequences starts at the same time. If the user chooses the pipeline and sample to analyze on the user interface, the sample input/output file paths and names in each module are stored into the database. When the user starts the analysis, the user interface creates the request XML, including the selected pipeline number and the XML is sent to the workflow-based middleware by consuming RESTful Web Services.

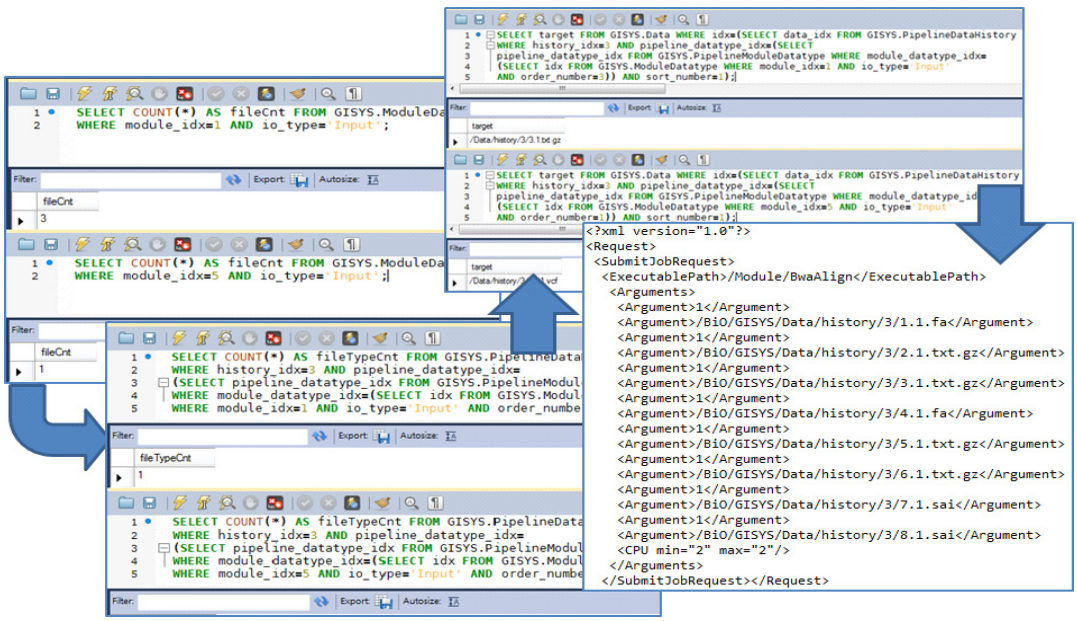
While the workflow-based middleware is analyzing the pipeline, it creates a log file which records the execution history of thepipeline. It also stores the analysis status information into the database and the information will be delivered to the user interfaces by Web services when the user requests the monitor information. If the analysis is successfully finished, the final DNA analysis information is stored into the database and the user is able to see the analysis results on the user interface by retrieving the information from the database. The queries in the DBMS of workflow-based middleware are to store pipeline, module andinput/output file information, and to update the running pipeline and module status information. The processes of these queries were described in the previous chapter and the query results for creating the request XML to the scheduler is explained in this chapter.

Figure 13 shows the queries for creating the request XML and its format for submitting a job to the scheduler. The workflow-based middleware calculates the number of file types for each input/output by a SELECT query. The



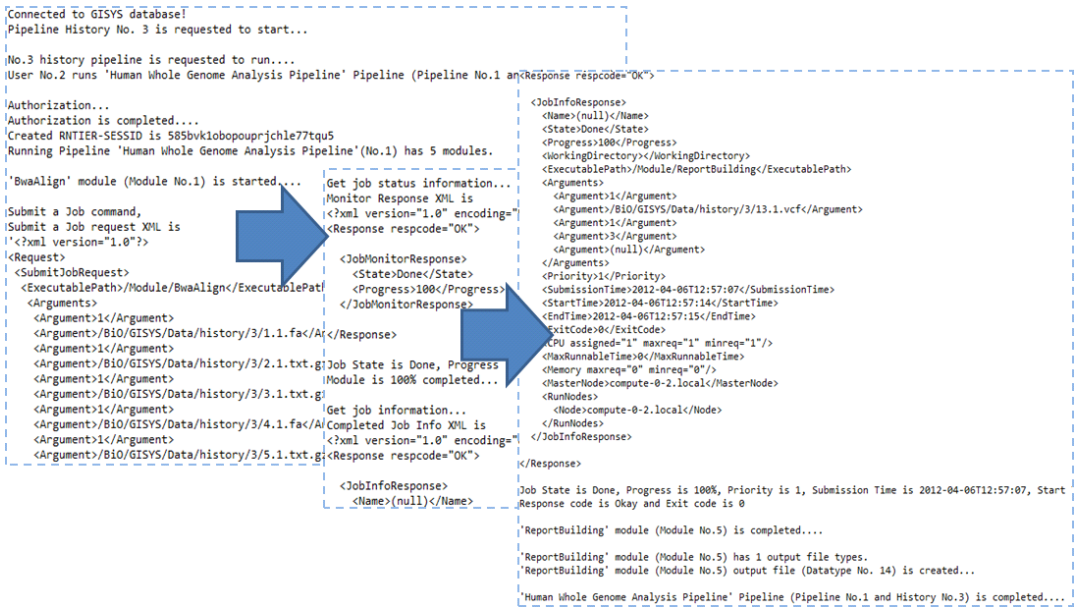
(그림 12) 워크플로우 기반 미들웨어 실행

(Figure 12) The example execution of workflow-based middleware



(그림 13) 워크플로우 기반 미들웨어에서의 데이터베이스 관리 화면

(Figure 13) The example database management in workflow-based middleware



(그림 14) 컴퓨팅 스케줄러 서버로부터의 웹 서비스 실행 절차
(Figure 14) The Web service consuming from computing scheduler servier

number of files in eachtype is computed by executing the same query and information is retrieved from each file, including file path and name. The command information is selected from the database and it is used with the input/output information for constructing the request XML file. The module command information is stored in the 'ExecutablePath' element and the input/output information becomes the value of the 'Argument' elements.

Figure 14 shows the Web service consuming results for the scheduler. If the pipeline analysis isstarted, the workflow-based middleware delivers the request XML to the scheduler for consuming job submission Web service. If a job is successfully submitted without errors, the scheduler conveys the response XML, including job id, to the workflow-based middleware. The responses of job status and job information Web service are also shown in the figure. The response information of job status is used for monitoring the module status and checking job completion. When the module is completely executed, the scheduler sends the job information including time, CPU, and memory numbers. Some of this information is stored into the database for monitoring the pipeline on the user-interface.

A Framework of Intelligent Middleware for DNA SequenceAnalysis in Cloud Computing Environment

5. CONCLUSION AND DISCUSSION

Scientific workflow techniques are applied in DNA sequence analysis systems because they provide the convenient way for complicated sequence analysis process. When analysts start analyzing, the system automatically runs commands or tools according to the analysis steps and they get the analysis result in a straightforward way by the workflow system. However, limited computing resources still pose problems in analysis process of a long time delay due to the massive DNA sequence information. Cloud computing is a concept for sharing computing resources in different computers and it has become one of the important technologies in the bioinformatics field. Since it reduces analysis time and helps users analyze a large amount of DNA sequences simultaneously, many system vendors try to introduce this advanced technology to their workflow systems. Also, Web service techniques are introduced for improving the interoperability between DNA sequence

analysis systems which are pre-built platforms and has the difficulty of embedding components.

As described above, various bioinformatics research vendors use scientific workflow systems and have made efforts to improve their systems for analyzing a large amount of DNA sequence data in a short amount of time. Most workflow system vendors consider Web services as key technologies for improving system performance. A national project is performed on the basis of two technologies in the bioinformatics field of Korea. The purpose of the project is to provide analysts that are not accustomed to computer-aid analysis tools, a fast and convenient platform for analyzing DNA sequences. The cloud computing is applied to the platform in order to utilize a large amount of computing resources in different supercomputing servers by creating lightweight virtual instances. The workflow-based middleware developed in this research is the part of the project for developing theplatform. The research is for developing a workflow engine which runs the DNA analysis pipeline and monitors the flow of the analysis. The significant functions of the workflow-based middleware are: 1) The workflow-based middleware runs a pipeline and monitors the status of each module in the pipeline using database management systems, 2) the requests and responses between the system components are performed on the basis of RESTful Web services. The workflow-based middleware contains components for parsing pipeline information and runs the pipeline following the flow script. It does not handle any complicated pipeline information in the memory of the system. Instead, a large amount of data are stored in the database and retrieved under necessity.

The workflow-based middleware improves the analysis performance by assisting in creating lightweight virtual instances since only minimum functions are included in the virtual instances and other heavy data is treated in the database at different systems. It is also easily able to be attached to various system components based on RESTful Web services with the XML data and simple URI. Since Web services provides convenient interfaces between systems that are developed in different languages under different platforms, they are used for developing a large system on the basis of various techniques. Many workflow systems introduce Web services in order to improve their interoperability. However, they use traditional Web services based on WSDL and SOAP. Since the traditional Web services have problems in implementation and in security by using SOAP protocol, RESTful Web services are newly introduced for improving existing services. It has advantages in building the services with simple URI and this technology is useful to improve the interoperability of the workflow engine. Therefore, RESTful Web services are adopted as the interfaces of the workflow-based middleware to the workflow designer, the user interfaces, and the schedulers. The research provides the lightweight workflow engine which has high interoperability with other system components in the cloud computing environment.

This research contributes to the development of a prototype of the workflow engine with two advanced technologies. It is confirmed that the analysis engine can analyze a large amount of DNA sequences in the cloud environment. However, the prototype is tested in a testbed environment and the environment has only one or two cloud instances and one user interface, one designer, and one scheduler. Therefore, a performance test based on a large amount of virtual instances must be performed. Also, more Web service functions will be added to the engine in order to support more designers and schedulers that have various pipeline modules and analysis commands.

참 고 문 헌(Reference)

[1] Genome Informatics System, http://gisys.kr, 2012.

[2] J. H. Song, K. H. Kim, “An XPDL-based Workflow Model Analyzer,” Review of Korean Society for Internet Information, Vol. 11, No. 6, pp.145-157, 2010.

[3] A. Kaya, “Workflow Interoperability: The WfMC Reference Model and an Imple-mentation,” Master Thesis, Technical University Hamburg-Harburg, Germany, 2001.

[4] D. Hoolingsworth, “The Workflow Reference Model: 10 Years On,” Fujitsu Services, United Kingdom, pp. 295-312, 2004.

[5] S. Kim, K. Yoon and Y. Kim, “A Design of Integrated Scientific Workflow Execution Environment for A Computational Scientific Application,” Reviewof Korean Society for Internet Information, Vol. 13, No. 1, pp.37-44,

2012.

[6] A. Barker and J. V. Hemert, “Scientific Workflow: A Survey and Research Direc-tions,” in Proceedings of the 7th international conference on Parallel processing and applied mathematics, pp. 746-753, 2007.

[7] Y. Hahn and S. Lee, “Bioworks, A scientific Workflow Platform for Problem Solving in Biological Domain,” in Proceedings of the 5th KOCON Conference, 2007.

[8] Kepler Project, https://kepler-project.org, 2009.

[9] B. Ludascher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Hones, E. A. Lee, J. Tao, and Y. Zhao, “Scientific Workflow Management and the KEPLER System,” Concurrency and Computation: Practice and Experience, Vol. 18, No. 10, pp. 1039-1065, 2006..

[10] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, and P. Li, “Taverna: a Tool for the Composition and Enactment of Bioinformatics Workflows,” Bioinformatics, Vol. 20, No. 17, pp. 3045-3054, 2004.

[11] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Greenwood, C. Goble, A. Wipat, P. Li, and T. Carver, “Delivering Web Service Coordination Capability to Users,” in Proceedings of the 13th international World Wide Web conference, pp. 438-439, 2004.

[12] Bioinformatics Workflow Builder Interface, http://www.alphaworks.ibm

[13] S. Lee, T. D. Wang, N. Hashmi and M. P. Cummings, Bio-STEER: A Semantic Web Workflow Tool for Grid Computing in the Life Science. Future Generation Computer Systems, 23, 3 (2007)

[14] I. Taylor, M. Shields, I. Wang, and R. Philp, “Distributed P2P Computing within Triana: A Galaxy Visualization Test Case,” in Proceedings of the 17th International Parallel and Distributed Processing Symposium, 2003.

[15] G. Allen, K. Davis, K. N. Dolkas, N. D. Doulamis, T. Goodale, T. Kielmann, A. Merzky, J. Nabrzyski, J. Pukacki, T. Radke, M. Russell, J. Shal, and I. Taylor, “Enabling Applications on the Grid: A GridLab Overview,” International Journal of High Performance Computing Applications: Special Issue on Grid Computing: Infra-structure and Applications, Vol. 17, No. 4, pp. 1-22, 2003.

[16] D. Blankenberg, G. V. Kuster, N. Coraor, G. Andanda, R. Lazarus, M. Mangan, A. Nekrutenko, and J. Taylor, “Galaxy: A Web-Based Genome Analysis Tool for Ex-perimentalists,” Informatics for Molecular Biologists, Vol. 19, pp. 1-21, 2010.

[17] S. Kim and Y. Park, “Overcoming limits of Bioinformatics using Cloud Computing,” Journal of KIISE: Computer Systems and Theory, Vol. 27, No. 6, pp. 27-34, 2009.

[18] L.D. Stein, “The case for cloud computing in genome informatics,” Stein Genome Biology, Vol. 11, p. 207, 2010.

[19] E. Deelman, D. Gannon, M. Shields, and I. Taylor, “Workflows and e-science: an Overview of Workflow System Features and Capabilities,” Future Generation Computer Systems, Vol. 25, No. 5, pp. 528-540, 2009.

[20] A.J. Nebro, G. Luque, F. Luna, E. Alba, “DNA fragment assembly using a grid-based genetic algorithm,” Computers & Operations Research, Vol. 35, pp. 2776-2790, 2008.

[21] Schadt, Eric E., et al., “Computational solutions to large-scale data management and analysis,” Nature Reviews Genetics, Vol. 11, No. 9, pp. 647-657, 2010.

[22] Geospiza, http://www.geospiza.com/cloud, 2010.

[23] Teragen Etax, http://www.thera-gen.com, 2011.

[24] KT ucloud, https://en.ucloudbiz.olleh.com/, 2010.

[25] Clunix supercomputing http://eng.clunix.com/, 2000.

◐ 저 자 소 개 ◑

오 준 석(Junseok Oh)
2002년 한성대학교 정보공학 학사
2004년 충북대학교 전자계산학 석사
2006년 The Pennsylvania State University MS
2010년 The Pennsylvania State University PhD
2011년~현재 연세대학교 연구교수
2013년~현재 경일대학교 특임교수
관심분야 : 클라우드컴퓨팅, 데이터과학, 분산컴퓨팅

이 윤 재(Yoonjae Lee)
2006년 연세대학교 전기전자공학 학사
2012년~현재 연세대학교 정보대학원 석사과정
관심분야 : 미래인터넷, 모바일네트워크, 방송통신융합정책

이 봉 규(Bong Gyou Lee)
1988년 연세대학교 경제학 학사
1992년 Cornell University MS
1994년 Cornell University Ph.D
1997년~2004년 한성대학교 정보전산학부 교수
2005년~현재 연세대학교 정보대학원 교수
2013년~현재 한국인터넷정보학회회장
관심분야 : IT 정책·산업, 방송통신융합정책, 모바일인터넷