International Research Journal in Global Engineering and Sciences. (IRJGES)
ISSN : 2456-172X | Vol. 2, No. 3, September - November, 2017 | Pages 46-51

DATA ORIENTED WORK FLOW MANAGEMENT FOR DISTRIBUTED DATA MINING APPLICATIONS

Dr.P.BANUMATHI Ms.T.SAKTHI SREE Ms. S.P.VIDHYA PRIYA
Associate Professor[1], Assistant Professor[2], Assistant Professor[3]
sakthisree@gmail.comspvidhya.priya@gmail.com bhanumathi_mohankumar@yahoo.com
Department of CSE, Kathir College of Engineering, Coimbatore, Tamil Nadu, India.

ABSTRACT

Workflow environments are used in data mining systems to lever data and execution flows associated to difficult applications. Weka, one of the frequently used open-source data mining systems, includes the Knowledge Flow environment which provide a drag-and-drop interface to create and perform data mining workflows. The Weka KnowledgeFlow permits users to execute workflow only on one computer. Most data mining workflows include several autonomous branches that could be run in parallel on a set of distributed machines to cut the overall execution time. I implemented distributed workflow implementation in Weka4WS, a framework that includes the facilities of Weka and its KnowledgeFlow environment to extend distributed resources accessible in a Grid using Web Service technologies. In this paper I illustrate the Weka4WS architecture and the functionalities provided by its service-oriented Knowledge Flow module, presenting its utilization to compose and execute easy parallel data mining workflows. Furthermore, I present ongoing work aimed at supporting also data-parallel workflows on a Grid.

1. Introduction

Data mining systems are used by lot of scientific and business organizations to learn helpful knowledge from the increasing amount of data they acquire. Workflow environments are commonly employed in data mining systems to manage data and execution flows associated to complex applications. One of the most used data mining systems is Weka, an open-source framework providing a wide set of algorithms and tools for processing and analyzing data.

The Weka toolkit includes a workflow environment called KnowledgeFlow, which provides a drag-and-drop interface to compose and execute data mining workflows referred to as "knowledge flows." A knowledge flow describes interactions and execution flows among data sources, filtering tools, data mining algorithms, and visualizers, allowing users to define and execute complex data mining applications. Once defined, knowledge flows can be stored and retrieved for modifications and/or re-execution: this allows users to define distinctive application patterns and reuse them in different contexts.

The Weka KnowledgeFlow allows users to carry out a complete workflow only on a single machine. On the other hand, most knowledge flows include several autonomous branches that could be run in parallel on a set of distributed machines to reduce the overall execution time. We implemented distributed workflow execution in Weka4WS [8], a framework that extendsWeka and its KnowledgeFlow environment to use distributed machines available in a Grid system using Web Service technologies.

The Grid facilities are exploited by Weka4WS because it provides a set of services to access distributed computing nodes, which can be effectively used to run complex and resource-demanding data mining applications [9]. In particular, Weka4WS adopts a service-oriented architecture in which Grid nodes rendering a wide set of data mining algorithms as Web Services, and client applications can

call up them to run distributed data mining applications defined as workflows.

In this paper I describe theWeka4WS architecture and the functionalities provided by its KnowledgeFlow component, showing its use to create and execute distributed data mining workflows on a Grid. In this paper I present ongoing work aimed at supporting data parallel workflows on a Grid.

2. Related Work

A few systems are related to Weka4WS for their focus on supporting distributed data mining workflows.

A service-oriented approach like Weka4WS is adopted by FAEHIM [1], which exposes a set of data mining algorithms as Web Services. However, in a diverse way from Weka4WS, FAEHIM does not provide a workflow system of its own, but relies on Triana [6] for composing data mining services as workflows. Furthermore, the FAEHIM services are not based on the WSRF technology.

The Knowledge Grid [2] is another service-oriented system supporting distributed data mining workflow execution. Like Weka4WS, it uses WSRF as enabling technology. UnlikeWeka4WS, which extends an already existing workflow system (the Weka KnowledgeFlow), the Knowledge Grid defines its own workflow formalism and provides a set of services to maintain the workflow execution.

Some other systems are related to Weka4WS for their support to distributed data mining on the Grid. In any case, Weka4WS differs from all of them for its focus on exploiting and extending the well-established Weka toolkit, thus allowing domain experts to create and execute distributed data mining workflows using a renowned user interface.

3.Weka4WS architecture

The novel Weka toolkit may be considered as it were made by two main parts: the Weka Library (WL) and the Graphical User Interface (GUI). The Weka Library is an extendible set of data mining algorithms for classification, clustering and association rules discovery. The Graphical User Interface provides a set of visual tools, which permit users to execute data mining algorithms on given datasets; it includes the Knowledge-Flow environment discussed earlier. At the same time Weka is logically composed by two separate parts, it is in fact a single application that runs on a single machine.

Weka4WS broaden Weka by extracting its WL part and deploying it as Web Service on a set of isolated Grid nodes [4]. Weka4WS also broaden the Weka GUI to allow the execution of data mining tasks both locally and remotely: the local execution is executed through the local WL, as in the original Weka system; the remote execution is performed by invoking one or more WLs presented as Web Services on the Grid. For the applications designed with the KnowledgeFlow, the data mining algorithms belonging to independent branches of the workflow can be executed in parallel on diverse Grid nodes.

The Web Services implemented in Weka4WS obey with the Web Services Resource Framework (WSRF), a set of standards related to the designing, addressing, inspection and lifetime management of stateful resources using Web Services. WSRF is widely adopted as the standard technology for implementing Grid services and systems. So, the use of WSRF as center technology allows Weka4WS to be easily incorporated with the most used Grid platforms, like Globus Toolkit [5].

Weka4WS has been developed using the WSRF libraries provided by Globus Toolkit and uses its services for standard Grid functionalities such as security and data transfer. The Weka4WS framework has two types of nodes:

- User node: the node where the client side of the application runs. It includes an extension of the Weka Graphical User Interface (GUI), the Weka Library (WL), and a Client Module (CM).
- Computing node: the server side of the application. It includes a Web Service (WS) and the Weka Library (WL). The WS can be

invoked by remote user nodes to present data mining tasks.

Data to be mined may be situated either at the user node, or at the computing node, or at some other remote resource (for example some shared repositories). When data are not available at the computing node they are moved by means of GridFTP [10], a reliable data transfer protocol which is part of Globus Toolkit.

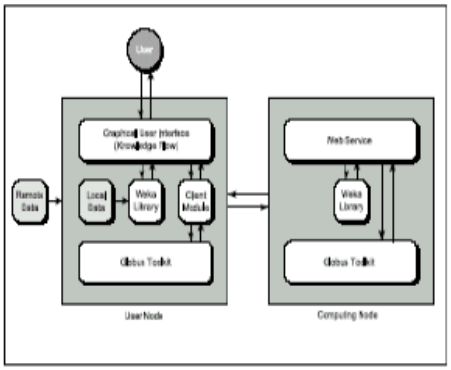

Figure 1. Schematical representation of the user and the computing nodes.

Figure 1 shows the Weka4WS software components of the user node and the computing node, and the interactions among them.

A user works on the KnowledgeFlow GUI to make a data mining workflow, typically composed by multiple data mining tasks: the local tasks are carried out by invoking the local WL, while the remote ones are carried out through the CM which acts as liaison between the GUI and the remote Web Services. Each task is performed in a thread of its own thus permitting to run multiple tasks in parallel.

At a computing node, the WS responses to the CM requests by invoking the suitable algorithms in the underlying WL. The calling upon of the algorithms is performed in an asynchronous way, i.e. the CM submits the task in a non-blocking mode and outcomes are either notified to it as soon as they are computed (push-style mode) or they are frequently checked for readiness by the client (pull-style mode) depending on the network configuration (e.g., the presence of a firewall).

4. KnowledgeFlow Weka4WS

KnowledgeFlow is a part of Weka, which permits composing workflows for giving out and analyzing data. A workflow can be completed by selecting components from a tool bar, inserting them on a layout canvas and linking them together: each component of the workflow is required to carry out a specific step of the data mining process.

Weka4WS broaden the Weka KnowledgeFlow permitting the execution of distributed data mining workflows on Grids by adding annotations into the knowledge flows. Through annotations a user can state how the workflow nodes can be mapped onto Grid nodes.

The selection of the location where to run a certain algorithm is made into the configuration panel of each algorithm, available by right clicking on the given algorithm and choosing Configure. A drop down menu has been added through which it is possible to select either the exact Grid host where the desired current algorithm to be executed (where local host will make the algorithm be computed on the local machine) or to let the system robotically choose one by selecting the auto entry. The presently used strategy in the auto mode is round robin: on each invocation the host in the list next to the previously used one is picked.

In a knowledge flow, each node representing a data-mining algorithm can be linked to a performance evaluator node, a component which evaluates the model generated by a data mining algorithm and makes a set of performance indices about that model. Although a data mining algorithm and its performance evaluator are depicted by two separate workflow nodes, the model generation and its evaluation are actually both performed in combination at the computing node chosen (or to be automatically chosen, in case the auto mode is selected) for the data mining algorithm.

The calculations may be started either by selecting the Start loading entry in the right-click context menu of each loader component of the flow (just like usually done in the conventional Weka Knowledge Flow) or by

clicking the Start all executions button in the right-top corner of the window (which is more convenient in flows with multiple loader components). Clicking a Log button, in the right-lower corner of the window, it is likely to follow the computations in their very single steps as well as to know their execution times.

4.1. Remote execution of workflow

Each data mining task defined in a workflow is carried out as a sequence of steps, explained in the following through an example. In this example we assume that the considered data mining task requests the execution of a classification task on a dataset which is stored at the user node, but not at the computing node. This is to be considered a worst scenario because in many cases the dataset to be mined is already available (or replicated) on the Grid node where the task is to be submitted.

The whole execution steps may be divided into 8 steps shown in Figure 2:

1.Resource creation: the create Resource process is invoked to create a new resource that will maintain its state throughout all the succeeding invocations of the Web Service until its destruction. The state is stored as properties of the resource. After the resource has been formed the Web Service returns the endpoint reference (EPR) of the created resource. Subsequent requests from the Client Module will be transferred to the resource identified by that EPR.

2. Notification subscription and notifications check: the subscribe operation is called in order to alert about changes that will occur to the Result resource property. As soon as this property value modifies (that is upon the conclusion of the data mining task) the Client Module is alerted of it. Just after the subscribe operation the not if Check operation is called to request the immediate send of a dummy notification to check whether the Client Module is able to receive notifications or not: in the first case the Client Module will employ the push-style mode, otherwise it will proceed in the pull-style mode.

3. Task submission: the classification operation is invoked in order to request for the execution of the classification task. The operation returns a Response object. If a replica of the dataset is not already available at the computing node, then the field datasetFound is set to false and the dirPath field is set to the URL where the dataset has to be uploaded; likewise, when a justification is required on a test set which is dissimilar from the dataset and the test set is not previously available at the computing node, the testsetFound field is set to false. The URL where the test set has to be uploaded is similar to that of the dataset.

4. File transfer: since in this example we understood that the dataset was not already available at the computing node, the Client Module requires to transfer it to the computing node. To that end, a Java GridFTP client [7] is instantiated and initialized to interoperate with the computing node GridFTP server: the dataset (or test set) is then moved to the computing node machine and stored in the directory whose path was specified in the dirPath field contained in the Response object returned by the classification operation;

5. Data mining: the Web Service begins the classification analysis through the invocation of the appropriate Java class in the Weka library. The outcomes of the computation are stored in the Result property of the resource created on Step 1;

6. Notification reception: Immediately after the Result property is changed a notification is sent to the Client Module by invoking its inherent deliver operation. This method allows the asynchronous delivery of the execution results as soon as they are created. In those cases where the client is unable to receive notifications the client will be checking from time to time the results for readiness through the value of the Result's field ready;

7. Results retrieving: the Client Module invokes the operation getResourceProperty in order to take back the Result property containing the results of the computation;

8. Resource destruction: The Client Module invokes the destroy operation, which removes the resource created on Step 1.

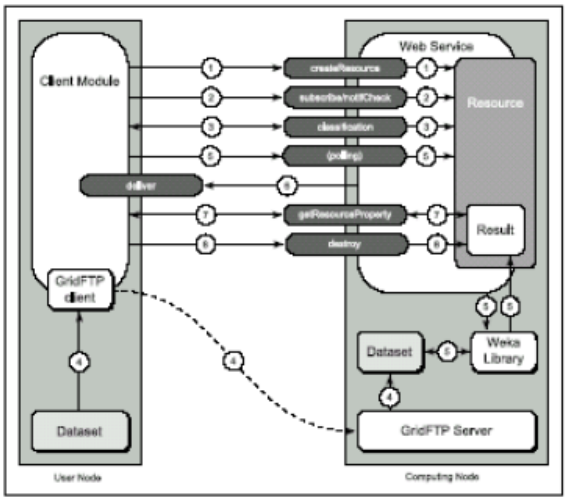

Figure 2. Execution mechanism of a single workflow task.

5. Data-parallel workflows

A type of parallelism, which runs multiple independent tasks in parallel on different processors, available on a single parallel machine or on a set of machines connected through a network like the Grid.

Another type of parallelism that could be efficiently exploited in data mining workflows is data parallelism [3], where a vast data set is cracked into smaller parts, each part is processed in parallel, and the results of each processing are then combined to produce a single result.

Both parallelism type aim to reach execution time speedup, and a better utilization of the computing resources, but while simple parallelism spotlights on running multiple tasks in parallel so that the execution time matches to the slowest task, data parallelism focuses on minimizing the execution time of a single task by dividing it into subtasks, each one operating on a subset of the original data.

The data-parallel approach is extensively employed in distributed data mining as it permits to process very large datasets that could not be analyzed on a single machine due to memory restrictions and/or computing time constraints. An example of data-parallel application is distributed classification: the dataset is divided into different subsets that are analyzed in parallel by multiple instances of a given classification algorithm; the resulting "base classifiers" are then used to get a global model through a variety of selection and combining techniques.

Our work is focusing on supporting data parallelism in the Weka4WS KnowledgeFlow to execute distributed data mining patterns like distributed classification. To hold this type of paradigm, we created a new component in the KnowledgeFlow, called DataSetPartitioner. A DataSetPartitioner component accepts one dataset in input, divides it into a number of partitions equal to the number of its outgoing arcs, and assigns each partition to one workflow node representing a data mining algorithm (for example, a classification algorithm). In case the data mining algorithm is annotated to run on a remote computing node, the subset assigned to it will be moved to that computing node.

6. Conclusions

Workflow management systems are of great significance to hold an effective design and execution of data mining applications. One of the main data mining workflow systems is the KnowledgeFlow environment of Weka. The Weka KnowledgeFlow permits users to execute a whole workflow only on a single machine. In this paper I presented a Grid-enabled version of Weka, called Weka4WS, which gives an addition to the KnowledgeFlow environment to support distributed execution of data mining workflows on a Grid.

Weka4WS employs a service-oriented approach in which all the Weka data mining algorithms are covered as Web Services and deployed on Grid nodes; users can create and invoke those services in a apparent way by defining data mining workflows as in the original Weka KnowledgeFlow. This approach permits to define simple parallel data mining applications in a simple and effective way.

I am currently working to make Weka4WS supporting also data parallelism. To this end, I implemented new components in the KnowledgeFlow which permit data partitioning

and model selection. These components allow to name a wider set of data mining applications, like distributed classification, to be planned and executed as workflows. I am planning a set of experiments to assess the scalability of various data-parallel workflows on large datasets.

References

1) A.Ali Shaikh , O. F. Rana, I. J. Taylor. Web Services Composition for Distributed Data Mining. Workshop on Web and Grid Services for Scientific Data Analysis, 2005.
2) A.Congiusta, D. Talia, P. Trunfio. Distributed data mining services leveraging WSRF. Future Generation Computer Systems, 23(1):34-41, 2007.
3) C.Pautasso, G. Alonso, Parallel Computing Patterns for Grid Workflows, Workshop on Workflows in Support of Large-Scale Science, 2006.
4) D.Talia, P. Trunfio, O. Verta. The Weka4WS framework for distributed data mining in service-oriented Grids. Concurrency and Computation: Practice and Experience, 20(16): 1933-1951, 2008.
5) I.Foster. Globus Toolkit Version 4: Software for service-oriented systems. Conference on Network and Parallel Computing, LNCS 3779: 2-13, 2005.
6) I.Taylor, M. Shields, I. Wang, A. Harrison. The Triana Workflow Environment: Architecture and Applications. In: I. Taylor, E. Deelman, D. Gannon, M. Shields (Eds.) Workflows for e-Science, Springer: 320-339, 2007.
7) Java GridFTP client. http://www.globus.org/cog/jftp [Visited: 14 January 2009]
8) M. Lackovic, D. Talia, P. Trunfio. Service Oriented KDD: A Framework for Grid DataMiningWorkflows. 10th International Workshop on High Performance Data Mining, 2008.
9) S. Hettich, S. D. Bay. The UCI KDD Archive, University of California, Department of Information and Computer Science. http://kdd.ics.uci.edu [Visited: 14 January 2009]
10) W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, I. Foster. The Globus striped GridFTP framework and server. Supercomputing Conference, 2005.