Service Oriented KDD: A Framework for Grid Data Mining Workflows

Marco Lackovic
Domenico Talia
Paolo Trunfio

Dept. of Electronics, Computer Science and Systems
University of Calabria, Italy
Goal of this work

• Design and implementation of a framework for composing and executing *data mining workflows* on the *Grid*

  ▪ *Data mining workflows*: allow to define typical/complex patterns and reuse them in different contexts

  ▪ The *Grid*: offers computing power, communication and data storage for the needs of complex DDM applications
The Weka4WS approach

• The workflow framework presented in this work is part of Weka4WS, a Grid-enabled version of Weka

• Weka4WS extends Weka to allow the remote execution of DM tasks on remote Grid nodes:
  - **Service oriented**: all the Weka algorithms are exposed as Web services
  - **Standard based**: Grid standards are used for authentication, authorization and file transfer (as provided by Globus Toolkit 4)
Workflow support in Weka4WS

- Weka4WS provides a Grid version of two Weka environments: Explorer and KnowledgeFlow
  - **Explorer**: an interface for data preprocessing, data mining and visualization
    - Implemented as part of a previous version of Weka4WS (Weka4WS 1.0)
  - **KnowledgeFlow**: a drag and drop interface which allows to compose and execute **data mining workflows**
    - Implemented in the version of Weka4WS presented here (Weka4WS 2.x)
Weka4WS architecture

- Local task
- Remote task
Weka4WS KnowledgeFlow

• Compared to the Weka version, the KnowledgeFlow of Weka4WS provides the following additional features:
  ▪ User authentication (Grid proxy generation)
  ▪ Selection of the location where each DM algorithm (or a set of DM algorithms) in the workflow has to be executed:
    • Local host
    • User-selected remote host
    • Automatically-selected remote host (round robin policy)
  ▪ Parallel execution of the different branches of the workflow
    • Multi-threaded: exploits both local (multi-processor, multi-core) and distributed parallelism
Weka4WS KnowledgeFlow: Screenshot

- Grid proxy generation
- Remote execution
- Start/stop control
Weka4WS KnowledgeFlow: Host selection

1) Workflow algorithm node configuration

2) Host selection
“Grouped” host selection
Weka4WS KnowledgeFlow

All branches start

Single branch start
Execution of a single workflow task

**Scenario:** a workflow node requires the execution of a clustering task on a dataset local to the user host.
1) **Resource creation.** The client invokes the `createResource` operation to create a new WS-Resource. A "Result" property is used to store the result of the clustering task. The WS returns the EPR of the created resource.
2) **Notification subscription.** The client invokes the *subscribe* operation, which subscribes to notifications about changes that will occur to the *Result* resource property.
3) **Task submission.** The client invokes the *clustering* operation. The operation returns a “Response” containing the information on whether the dataset is present at the computing node and the directory where to upload it in case it is not present.
Dataset file transfer. The client establishes a connection with the GridFTP Server at the computing node and sends the dataset to the location specified in the Response returned from the clustering invocation.
5) **Data mining**. The clustering analysis is started by invoking the appropriate Java class in the Weka library. The execution is handled within the WS-Resource created on Step 1, and the result of the computation is stored in the **Result** property.
6) Results delivery. As soon as the Result property has been changed, its new value is notified to the client, by invoking its implicit deliver operation.
7) **Resource destruction**. The client invokes the `destroy` operation, which explicitly destroys the WS-Resource created on Step 1.
Results delivery: push style

The request reaches the service via the firewall tunnel opened to the computing node.

The notification reaches the user node.

The notification reaches the user node.
**Results delivery: pull style**

The request reaches the service via the firewall tunnel opened to the computing node.

The notification is blocked by the user node NAT Router/Firewall.
Example: clustering workflow

• A simple workflow in which a dataset is analyzed in parallel using five instances of the same clustering algorithm with different configurations
  - Datasets: covertype (100k, 200k, 400k instances)
  - Algorithm: KMeans
  - Configurations: 3 to 7 clusters to be found
Example: clustering workflow

- Performance results on five Grid nodes:
  - Machines configurations: 2.8-3.2 GHz CPUs, 1-2 GB RAM, belonging to two different LANs.
  - Execution speedup ranged from 2.13 to 2.67
Conclusion

• The Weka4WS KnowledgeFlow allows the parallel execution of the DM algorithms which are part of the workflow, hence allowing to reduce execution time.

• Future developments:
  ▪ Efficient assignment of the workflow tasks to the available Grid nodes, based on their current status
  ▪ Support distribution of the data across different parallel computing nodes (data parallelism)
Thanks for your attention!

Weka4WS is open source and freely available at:

http://grid.deis.unical.it/weka4ws
Extra slide: Weka4WS Explorer