Scalable Data Mining on Parallel, Distributed and Cloud Computing

Domenico Talia
DIMES, Università della Calabria & DtoK Lab
Italy
talia@dimes.unical.it

Lecture Goals

- Parallel systems, distributed computing infrastructures, like Grids and P2P networks, and Cloud computing platforms, offer an effective support for addressing both the
  - computational and
  - data storage needs
  of Big Data mining and parallel analytics applications.

- Complex data mining tasks involve data- and compute-intensive algorithms that require large storage facilities together with high performance processors to get results in suitable times.
Lecture Goals

- In this lecture we introduce the most relevant topics and the main research issues in high performance data mining including:
  - parallel data mining strategies,
  - distributed analysis techniques, and
  - knowledge services and Cloud-based data mining.

- We discuss parallel models, scalable data mining algorithms, data mining programming tools and applications.

General Outline

LESSONS

1. Parallel Data Mining Strategies

2. Distributed and Service-oriented Data Mining

3. Data Mining Tools on Clouds
LESSON 2

Distributed and Service-oriented Data Mining

Parallel and Distributed Data Mining

- Parallel data mining
  - Task or control parallelism
  - Independent parallelism
  - SPMD parallelism
  - Hybrid parallelism

- Distributed data mining
  - Voting
  - Meta-learning
  - Ensemble learning
  - Collective data mining

Can be a component of
Data Mining and Computational Needs

- Today data is **often distributed** geographically or locally. Data come from the Internet, from sensors, from mobile devices, from the sky, from distributed DBMS.
- When
  - **large data sets** are coupled with
  - **geographic distribution** of data, users, and systems,
  - it is necessary to combine different technologies for implementing **high-performance distributed knowledge discovery systems**.

---

Talk outline

- Distributed Data Mining
- Meta-Learning
- Collective data mining
- Ensemple learning
- Distributed Mining systems
- Service-oriented data mining
- Weka4WS, Knowledge Grid, mobile data mining.
Distributed Data Analysis Patterns

- Data parallelism? SPMD or Task parallelism?
- Managing data dependencies
- Dynamic task graphs/workflows (data dependencies)
- MapReduce, MPI, OpenMP, Web Services, Java, ...
- Dynamic data access involving large amounts of data
- Parallel data mining and/or Distributed data mining
- Programming distributed mining operations/tasks/patterns

---

Programming Levels

Grain size
- Web Services, Grid Services, Workflows, Mashup, ...
- Components, Patterns, Distributed Objects, ...
- MPI, OpenMP, threads, MapReduce, RMI, HPF, ...

Process #
Distributed Data Mining

- Typically, in distributed data mining algorithms the same code runs on each data site concurrently, producing one local model per site.
- Subsequently, all local models are aggregated to produce the final model.

The most important (and used) distribute data mining strategies are
  - Ensemble learning,
  - Meta-learning, and
  - Collective mining.
Meta-Learning

- Meta Learning can be defined as learning from learned knowledge.

- The Meta Learning techniques aim at implementing a global model from a set of local models coming from some distributed data sets.

- In a data classification scenario, a global model is obtained by learning from the predictions of a set of base classifiers on a common validation set.
  1. In the first step a set of classifiers are generated by applying a data mining algorithm on a collection of data sets.
  2. Then the classifiers are combined in a single site to produce a global classifier.

Meta-Learning

- A meta-class attribute indicates the best algorithm for the problem, among a set of candidate algorithms.

- The class label stored in a meta-example is usually defined via a cross-validation experiment using the available problem’s dataset.

- The meta-learner in this case is simply a classifier which predicts the best algorithm for a given problem based on its descriptive meta-attributes.
Meta-Learning

- The steps needed to build a global classifier from a set of distributed training sets according to a meta-learning approach:
  1. The initial training sets are used as input of \( N \) learning algorithms that in parallel build \( N \) classification models (base classifiers).
  2. A meta-level training set is built by combining the predictions of the base classifiers on a common validation set.
  3. A global classifier is trained from the meta-level training set by a meta-learning algorithm.

Stacking and Boosting

- **Stacking** is a way of combining multiple heterogeneous models in meta-learning.
- Stacking is not used to combine models of the same type. It is applied to models built by using different learning algorithms.
- It does not use a voting approach, but tries to learn which classifiers are the reliable ones, using another learning algorithm (the meta-learner) to discover how best to combine the output of the base learners.

- In **Boosting** techniques a weight is assigned to the different models produced on the basis of the errors of each model on the training data.
Collective Data Mining

- **Collective data mining** does not combine a set of complete models generated at each site on partitioned or replicated data sets,
  
  But
  
  It builds the global model through the combination of partial models computed in the different sites.

- The global model is directly composed by summing an appropriate set of basis functions.

Collective Data Mining

- The global function $f(x)$ that represents the global model can be expressed as:

  $$ f(x) = \sum w_k \psi_k(x) $$

  where $\psi_k(x)$ is the $k$-th basis function and $w_k$ is the corresponding coefficient that must be learned locally on each site based on the stored data set.

- This result is founded on the fact that any function can be expressed in a distributed fashion using a set of appropriate basis functions that may contain nonlinear terms.
Collective Data Mining

Main steps of the collective data mining approach:

1. Select an appropriate orthonormal representation for the type of data model to be generated.

2. Generate at each site approximate orthonormal basis coefficients.
   2.1 If the global function includes nonlinear terms, move a sample of data sets from each site to a central site and generate there the approximate basis coefficients corresponding to such nonlinear terms.

3. Combine the local models to generate the global model and transform it into the user described model representation.

Ensemble Learning

- An ensemble learning method builds a set of base classifiers from training data and performs classification
  - by voting (in the case of classification) or
  - by averaging (in the case of regression) on the predictions made by each classifier.

- The final result is the ensemble classifier, which very often have higher classification quality than any single classifier that has been used to compose it.
Ensemble Learning (Democratic Learning)

- These are the main steps that compose an ensemble learning strategy for data classification:
  1. Using a partitioning tool the input data set is split into a training set and a test set.
  2. The training set is given in input to $N$ classification algorithms that run concurrently on different processing nodes to build $N$ independent classification models.
  3. Then, a voter tool $V$ accesses the $N$ models and performs an ensemble classification by assigning to each instance of the test set the class predicted by the majority of the $N$ models generated at the previous step.

Ensemble Learning

- The identification of optimal ways to combine the base classifiers is a crucial point here.

- The most adopted approaches are two schemes called bagging and boosting.

- Bagging (called voting for classification and averaging for regression) combines the predicted classifications (prediction) from multiple models.

- In bagging, the models receive equal weight, whereas in boosting, weighting is used to give more influence to the more successful ones.
Distributed data mining: Papyrus

- **Papyrus** is a distributed data mining system developed for clusters and super-clusters of workstations as composed four software layers:
  - data management,
  - data mining,
  - predictive modeling, and
  - agent.

Distributed data mining: Papyrus

- The data management layer is implemented as a global data warehouse that allows to move data from node to node.
- The agent layer selects strategies and resources and move predictive models among the cluster nodes.
- Thus this system has the ability to combine data and models movement.
- Papyrus is based on mobile agents implemented using Java aglets.
Distributed data mining: PaDDMAS

- PaDDMAS is a component-based tool set that integrates pre-developed or custom packages (that can be sequential or parallel) using a dataflow approach.

- PaDDMAS provides three types of components:
  - data management components,
  - data analysis components and
  - data visualization components.

Distributed data mining: PaDDMAS

- Each component is wrapped as a Java object with its interface specified in XML.

- Components can be located at distributed nodes.

- Connectivity to databases is provided thorough JDBC bridges.
Distributed data mining: KEDM

- Kensington Enterprise data mining is a PDKD system based on a three-tier client/server architecture.
- The three tiers include:
  - client,
  - application server and
  - third-tier servers (RDBMS and parallel data mining service).
- The data models (knowledge) built on a remote server are moved, under the control of the application server, to the client node for evaluation.

Distributed data mining: KEDM

- The Kensington system has been implemented in Java and uses the Enterprise JavaBeans component architecture.
- C+MPI data mining packages can be integrated in the Java framework.
- Databases located on the Internet can be accessed via a JDBC connection.
Distributed data mining: JAM

- JAM is an agent-based distributed data mining system developed to mine data stored in different sites for building so called meta-models.

- A meta-model is a combination of several models learned at the different sites where data are stored (Meta-learning).

Distributed data mining: JAM

- JAM uses Java applets to move data mining agents to remote sites.

- Knowledge discovery is speeded up by
  - executing in parallel a number of data mining processes on different data subsets and
  - combining the results through meta-learning.
Distributed data mining: BODHI

- Collective data mining, is implemented in the BODHI system.

- BODHI is implemented in Java as a hierarchy of four main components:
  - the individual agents,
  - the agent stations on each processing node,
  - the facilitator for coordination of agent stations and
  - the user interface for system configuration and control.

Distributed data mining: BODHI

- Like JAM and the Kensington system, BODHI supports the migration of mining agents over the nodes where data are stored.

- Produced models are collected on a central node where models are combined and analyzed.
Services for distributed data mining

-Exploiting the SOA model it is possible to define **basic services for supporting distributed data mining tasks/applications**.

-Data analysis in large scale distributed systems for science and industry (for example: from a private Cloud to InterClouds).

-Those services can address all the aspects of data mining and in knowledge discovery processes
  - data selection and transport services,
  - data analysis services,
  - knowledge models representation services, and
  - knowledge visualization services.

Collection of Services for Distributed Data Mining

- It is possible to design services corresponding to

<table>
<thead>
<tr>
<th>Data Mining Applications or KDD processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>This level includes the previous tasks and patterns composed in a multi-step workflow.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Distributed Data Mining Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>This level implements, as services, patterns such as collective learning, parallel classification and meta-learning models.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Single Data Mining Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Here are included tasks such as classification, clustering, and association rules discovery.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Single KDD Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>All steps that compose a KDD process such as preprocessing, filtering, and visualization are expressed as services.</td>
</tr>
</tbody>
</table>
Data mining services

- By exploiting the Web/Grid/Cloud services features it is possible to develop data mining services accessible every time and everywhere (remotely and from small devices).

- This approach can result in
  - Service-based distributed data mining applications
  - Data mining services for communities/virtual organizations.
  - Distributed data analysis services on demand.
  - A sort of knowledge discovery eco-system formed of a large numbers of decentralized data analysis services.

Data mining services

- Service-based systems
  - Weka4WS
  - KNOWLEDGE GRID
  - Mobile Data Mining Services
Weka4WS

The Weka4WS framework

- **Weka** is one of the most used open source suite for data mining.
- In Weka, the overall data mining process takes place on a single machine; the algorithms can be only locally executed.
- **Weka4WS** extends Weka to support distributed execution of the Weka data mining algorithms
  - All data mining algorithms provided by the Weka library are exposed as a stateful Web Service
  - Globus Toolkit 4 is used for basic Grid functionalities such as security and data transfer.

Weka4WS Architecture

- We distinguish Weka4WS nodes in two categories:
  - user nodes, which are the local machines of the users providing the Weka4WS client software
  - computing nodes, which provide the Weka4WS Web Services allowing the execution of remote data mining tasks

- Data can be located on computing nodes, user nodes, or third-party nodes

- If the dataset to be mined is not available on a computing node, it can be copied or replicated by means of the Globus data management services.

Software components

- User nodes include three software components:
  - Graphical User Interface (GUI)
  - Client Module (CM)
  - Weka Library (WL)
Computing nodes include two software components:

- Web Service (WS)
- Weka Library (WL)

The GUI extends the Weka Explorer environment to allow the execution of both local and remote data mining tasks:

- local tasks are executed by directly invoking the local WL
- remote tasks are executed through the CM, which operates as an intermediary between the GUI and Web Services on remote computing nodes
The WS is a Web Service that exposes the data mining algorithms provided by the underlying Weka Library.

Therefore, requests to the WS are executed by invoking the corresponding WL algorithms.

Web Services operations

- The first three operations are related to service-specific invocation mechanisms.
- The last three operations are used to require the execution of a specific data mining task.
**Web Services operations**

- The **classification** operation provides access to the complete set of classifiers in the Weka Library (currently, **more than 70** algorithms)

- The **clustering** and **associationRules** operations expose all the clustering and association rules algorithms provided by the Weka Library (**5** and **2** algorithms, respectively)

- To improve concurrency the data mining operations are invoked in an asynchronous way:
  - the client submits the execution in a non-blocking mode, and results are notified to the client whenever they have been computed

**Weka4WS Graphical User Interfaces**

- **Weka4WS** extends the GUIs of Weka:
  - **Explorer**
  - **KnowledgeFlow**
    - Both available with Weka4ws 2.1 ([grid.deis.unical.it/weka4ws](grid.deis.unical.it/weka4ws))
A "Control panel" allowing to submit both local and remote tasks has been added to the original Weka Explorer environment.

A drop down menu allows to choose where to run the current data mining task ("Local", "Auto", or a specific host).
Weka4WS Explorer

- Each task in the GUI is managed by an independent thread. A user can start multiple data mining tasks at the same time on different remote hosts.

Weka4WS Explorer

- Whenever the output of a data mining task has been received from a remote computing node, it is visualized in the standard Output panel.
Weka4WS KnowledgeFlow

- A data mining workflow can be composed and run on several Grid nodes

Weka4WS KnowledgeFlow

Remote execution
Weka4WS: Application speedup

The covertype dataset from the UCI archive has been used as data source. The dataset has a size of about 73 MB and contains information about forest cover type for 50000 sites in the United States. Each dataset instance corresponding to a site observation is described by 54 attributes that give information about the main features of a site (e.g., elevation, aspect, slope, etc.). The 55th attribute contains the cover type, represented as an integer in the range 1 to 7.

Weka4WS has been used to run an application in which 8 independent instances of the Kflume algorithm [17] perform a different clustering task on the covertype dataset. In

Knowledge Grid
The Knowledge Grid

- Knowledge Grid: a distributed knowledge discovery architecture that can be configured on top of generic Grid middleware
- A first prototype has been implemented on Globus based on a high-level user interface for application composition (VEGA)
- The Knowledge Grid services are currently being re-implemented as stateful Web Services.

Service-oriented Knowledge Grid

Service-oriented Knowledge Grid

- The Knowledge Grid operations have been re-implemented as stateful Web Services.
- They can be invoked by client interfaces, programs, and other services.

The Knowledge Grid Services

- Each K-Grid service is exposed as a Web Service that exports one or more operations.
- The operations exported by the High-level K-Grid services are invoked by user-level applications.
- The operations provided by the Core K-Grid services are invoked both by High-level and Core K-Grid services.

Knowledge Grid: Service operations

<table>
<thead>
<tr>
<th>Service</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAS</td>
<td>publishData</td>
<td>This operation is invoked by a client for publishing a newly available dataset. The publishing requires a set of information that will be stored as metadata in the local KMR.</td>
</tr>
<tr>
<td></td>
<td>searchData</td>
<td>Data to be used in a KDD computation is located during the application design by invoking this operation. The searching is performed on the basis of appropriate parameters.</td>
</tr>
<tr>
<td>TAAS</td>
<td>publishTools</td>
<td>This operation is used to publish metadata about a data mining tool in the local KMR. As a result of the publishing, a new DM service is made available for utilization in KDD computations.</td>
</tr>
<tr>
<td></td>
<td>searchTools</td>
<td>It is similar to the searchData operation except that it is targeted to data mining tools.</td>
</tr>
</tbody>
</table>
**Knowledge Grid: Service operations**

<table>
<thead>
<tr>
<th>Service</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPMS</td>
<td>submitApplication</td>
<td>This operation receives a conceptual model of the application to be executed. The EPMS generates a corresponding abstract execution plan and submits it to the RAEMS for its execution.</td>
</tr>
<tr>
<td>RPS</td>
<td>getResults</td>
<td>Retrieves results of a performed KDD computation and presents them to the user.</td>
</tr>
<tr>
<td>KDS</td>
<td>publishResource</td>
<td>This is the basic, core-level operation for publishing data or tools. It is thus invoked by the DAS or TAAS services for performing their own specific operations.</td>
</tr>
<tr>
<td></td>
<td>searchResource</td>
<td>The core-level operation for searching data or tools.</td>
</tr>
<tr>
<td>RAEMS</td>
<td>manageExecution</td>
<td>This operation receives an abstract execution plan of the application. The RAEMS generates an instantiated execution plan and manages its execution.</td>
</tr>
</tbody>
</table>

**S-O Distributed Data Mining Workflows**

- **DIS3GNO** is a visual framework for programming and running service-oriented data mining workflows in the KNOWLEDGE GRID.

- **DIS3GNO** supports all the phases of a distributed knowledge discovery process, including composition, execution, and results visualization.
S-O Distributed Data Mining Workflows

- A data mining workflow is a graph in which
  - nodes typically represent data sources, filtering tools, data mining algorithms, and visualizers, and
  - edges represent execution dependencies among nodes.

- DIS3GNO supports all the phases of a distributed knowledge discovery process, including composition, execution, and results visualization.

- Each node is a service.

S-O Distributed Data Mining Workflows

- The workflow concept plays a fundamental role in the KNOWLEDGE GRID at different levels of abstraction.

- A client application submits a distributed data mining application to the KNOWLEDGE GRID by describing it through an XML workflow formalism (conceptual model).

- The conceptual model describes data and tools to be used, with or without specifying information about their location or implementation.
DIS3GNO: The Knowledge Grid GUI

- The Knowledge Grid user and programming interface is composed by 4 panels.

DIS3GNO: A Visual Framework

- DIS3GNO is the user front-end for two main KNOWLEDGE GRID operations:
  - *Metadata management*. DIS3GNO provides an interface to publish and search metadata about data and tools.
  - *Design and Execution management*. DIS3GNO provides an environment to design and execute distributed data mining applications as workflows, through the interaction with the execution services of the KNOWLEDGE GRID.
DIS3GNO: The Knowledge Grid GUI

- The state of a node is represented by a different icon

1. No information provided yet
2. Information incomplete
3. Resource unavailable
4. Ready to run
5. Running
6. Task completed

Workflow composition in DIS3GNO

- 5 editing modes

3. The algorithm has been selected
Searching resources

- Example of dataset search

1. Input of the searching criteria
2. Search process
3. Results selection

DIS3GNO: A Visual Framework

Programming a data mining workflow as a graph of services and run them in parallel.
DIS3GNO: A Visual Framework

Workflow running and results visualization after workflow completion.

Data Mining Workflows with DIS3GNO

Eight similar classifiers in parallel produce different classifications (using different parameters) of the same dataset. The best classification is selected by the ModelChooser node.
Performance Results

Execution time and speedup with different dataset sizes. With the 36 MB dataset, time is reduced from 21 hours to 3.5 hours.

Data Mining Workflows with DIS3GNO

In an ensemble learning application four different classifiers in parallel produce 4 classifications from 4 different training sets. The best classification is selected by voting.
Performance Results

Execution time and speedup with different dataset sizes. The overall execution time is bound to the execution time of the slowest algorithm, thus limiting the total speedup.

Mobile Data Mining Grid Services
Services for Mobile Data Mining

- The main research goal is to support a user to access data mining services on mobile devices.
- The system includes three components:
  - Data providers
  - Mining servers
  - Mobile clients

The Mining server

- A Mining server implements two Grid Services:
  - Data Collection Service (DCS): invoked by a data provider to store data in the data store.
  - Data Mining Service (DMS): invoked by a mobile client to ask for the execution of a data mining task.

Services for Mobile Data Mining

- A user can select which part of a result (data mining model) he wants to visualize.

Impact of Services overhead

### Execution times

- It can be observed that the data mining phase takes approximately from 95% to 99% of the total execution time.
- Thus the overhead due to the WSRF invocation mechanisms is negligible for typical data mining tasks on large datasets.
Impact of the Services overhead

Execution times

- In a larger Grid scenario the data mining step represents from 85% to 88% of the total execution time, the dataset download takes about 11%, while the other steps range from 4% to 0.5%

Final Remarks

- We are much more able to store data than to extract knowledge from it.

- Scientific and industrial applications must be able to analyze very large data sources (archives, databases, flat files).

- HPC, Grid and Cloud systems may be used as distributed infrastructures for service-oriented data mining applications.

- New HPC infrastructures allow us to attack new problems, BUT require to solve more challenging problems.
Final Remarks

- New models, frameworks, and environments are required
  - Data is becoming a BIG player, programming data analysis applications and services is a must.
  - New ways to efficiently compose different models and paradigms are needed.
  - The service-oriented approach can be a viable integration paradigm.

- In a long-term vision, pervasive collections of data analysis services and applications must be accessed and used as public utilities over the Internet.

Thanks

Questions?