Scalable Data Mining on Parallel, Distributed and Cloud Computing Systems

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 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

Data Mining Tools on Clouds

Talk outline

- Big problems and Big data
- Using Clouds for data mining
- A collection of services for scalable data analysis
- Data mining workflows
  - Data Mining Cloud Framework (DMCF)
  - JS4Cloud for programming service-oriented workflows.
- Final remarks.
Goals (1)

- KDD and data mining techniques are used in many domains to extract useful knowledge from big datasets.

- KDD applications range from
  - Single-task applications
  - Parameter-sweeping applications/ regular parallel applications
  - Complex applications (Workflow-based, distributed, parallel).

- Cloud Computing can be used to provide developers and end-users with computing and storage services and scalable execution mechanisms needed to efficiently run all these classes of applications.

Goals (2)

- How using Cloud services for scalable execution of data analysis workflows.

- A programming environment for data analysis: Data Mining Cloud Framework (DMCF).

- A visual programming interface VL4Cloud and the script-based JS4Cloud language for implementing service-oriented workflows.

- Evaluating the performance of data mining workflows on DMCF.
Data analysis as a service

- **PaaS** (Platform as a Service) can be an appropriate model to build frameworks that allow users to design and execute data mining applications.

- **SaaS** (Software as a Service) can be an appropriate model to implement scalable data mining applications.

- Those two cloud service models can be effectively exploited for delivering data analysis tools and applications as services.

Service-Oriented Data Mining

- Knowledge discovery (KDD) and data mining (DM) are:
  - Compute- and data-intensive processes/tasks
  - Often based on distribution of data, algorithms, and users.

- Large scale service-oriented systems (like Clouds) can integrate both distributed computing and parallel computing, thus they are useful platforms.

- They also offer
  - security, resource information, data access and management, communication, scheduling, SLAs, …
Services for Distributed Data Mining

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

- 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.

---

Services for Distributed Data Mining

- It is possible to design services corresponding to

  - **Data Mining Applications and KDD processes**
    - This level includes the previous tasks and patterns composed in *multi-step workflows.*

  - **Distributed Data Mining Patterns**
    - This level implements, as services, patterns such as *collective learning,* *parallel classification* and *meta-learning models.*

  - **Single Data Mining Tasks**
    - Here are included tasks such as *classification, clustering,* and *association rules discovery.*

  - **Single KDD Steps**
    - All steps that compose a KDD process such as *preprocessing,* *filtering,* and *visualization* are expressed as services.
Services for Distributed Data Mining

- This collection of data mining services implements an

![Open Service Framework for Distributed Data Mining]

- It allows developers to program distributed KDD processes as a **composition of single and/or aggregated services** available over a service-oriented infrastructure.

![A Service-oriented Cloud workflow]

- Those services should exploit other basic Cloud services for data transfer, replica management, data integration and querying.
Services for Distributed Data Mining

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

- This approach can produce not only service-based distributed data mining applications, but also
  - 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.

The Data Mining Cloud Framework

- Data Mining Cloud Framework supports workflow-based KDD applications, expressed (visually and by a script language) as a graphs that link together data sources, data mining algorithms, and visualization tools.
Architecture Components

- **Compute** is the computational environment to execute Cloud applications:
  - *Web role*: Web-based applications.
  - *Worker role*: batch applications.
  - *VM role*: virtual machine images.

- **Storage** provides scalable storage elements:
  - *Blobs*: storing binary and text data.
  - *Tables*: non-relational databases.
  - *Queues*: communication between components.

- **Fabric controller** links the physical machines of a single data center:
  - *Compute* and *Storage* services are built on top of this component.

The Data Mining Cloud Framework: Architecture
The Data Mining Cloud Framework - Mapping

The Data Mining Cloud Framework – Execution
Example applications (1)

**Finance:** Prediction of personal income based on census data

**E-Health:** Disease classification based on gene analysis

**Networks:** Discovery of network attacks from log analysis.

Example applications (2)

**Biosciences:** Drug metabolism associations in pharmacogenomics.

**Smart City:** Car trajectory pattern detection applications.
The Cloud4SNP workflow

- **DMET** (Drug Metabolism Enzymes and Transporters) has been designed specifically to test drug metabolism associations in pharmacogenomics case-control study.
- Cloud4SNP is a Cloud implementation of the DMET Analyzer by using the DMCF.

**Data Mining Cloud Framework**

DMET Microarray

- **Pharmacogenomics** is the science that allows us to predict a response to drugs based on an individual’s genetic makeup and searches for correlations between gene expression or Single Nucleotide Polymorphisms (SNPs) of patient’s genome and the toxicity or efficacy of a drug.

- SNPs data, produced by microarray platforms, need to be preprocessed and analyzed in order to find that correlation.

- The DMET (Drug Metabolism Enzymes and Transporters) platform comprises 225 genes involved in Absorption, Distribution, Metabolism and Excretion (ADME) of drugs.
DMET - Case Control - Data Analysis workflow

DMET-Analyzer Scalability Issues

- Parameter sweeping application.
- Researchers often need that data sets should be analyzed in parallel by multiple instances of the same data mining algorithm with different parameters.
- Graphical-based workflow applications: Researcher need support in application design.
- Computational Aspects: Running Times and Scalability.
Cloud4SNP workflow (execution)

Cloud4SNP workflow (final result)
Performance evaluation

![Graph 1](image1)

![Graph 2](image2)

Performance evaluation

![Graph 3](image3)
### Trajectory Pattern Detection

- Analyze trajectories of mobile users to discover movement patterns and rules.
- A workflow that integrate frequent regions detection, trajectory data synthesis and trajectory pattern extraction.

#### Data Mining Cloud Framework

![Data Mining Cloud Framework Diagram]

#### Application Main Steps

**Frequent Regions Detection.**
- Detect areas more densely passed through
- Density-based clustering algorithm (DB-Scan)
- Further analysis: movement through areas

**Trajectory Data Synthesis.**
- each point is substituted by the dense region it belongs to.
- trajectory representations is changed from movements between points into movements between frequent regions

**Trajectory Pattern Extraction.**
- Discovery of patterns from structured trajectories
- T-Apriori algorithm, i.e. ad-hoc modified version of Apriori
Application Workflow

Workflow Implementation

- Workflow implementing the trajectory pattern detection algorithm
  - Each node represents either a data source or a data mining tool
  - Each edge represents an execution dependency among nodes
  - Some nodes are labeled by the array notation
    - Compact way to represent multiple instances of the same dataset or tool
    - Very useful to build complex workflows (data/task parallelism, parameter sweeping, etc.)

128 parallel tasks!
Experimental Evaluation

**Turnaround time**

- vs the number of servers (up to 64), for different data sizes
- vs several data sizes (up to 128 timestamps), for different number of servers

- comparison parallel/sequential execution
- $O_{16}(O_{128})$: it reduces from 8.3 (68) hours to about 0.5 (1.4) hours
- it proportionally increases with the input size
- it proportionally decreases with the increase of computing resources

---

**Experimental Evaluation**

- **Scalability indicators**
  - speed-up
  - notable trend, up to the case of 16 nodes
  - good trend for higher number of nodes (influence of the sequential steps)
  - scale-up
  - comparable times when data size and #servers increase proportionally
  - DBSCAN step (parallel) takes most of the total time
  - other steps (sequential) increases with larger datasets
Script-based workflows

- We extended DMCF adding a **script-based data analysis programming model** as a more flexible programming interface.

- Script-based workflows are an effective alternative to graphical programming.

- A script language allows experts to program complex applications more rapidly, in a **more concise** way and with **higher flexibility**.

- The idea is to offer a script-based data analysis language as an **additional and more flexible programming interface** to skilled users.

The JS4Cloud script language

- **JS4Cloud** (*JavaScript for Cloud*) is a language for programming data analysis workflows.

- Main benefits of JS4Cloud:
  - it is based on a well known scripting language, so users **do not have to learn a new language** from scratch;
  - it implements a **data-driven task parallelism** that automatically spawns ready-to-run tasks to the available Cloud resources;
  - it exploits **implicit parallelism** so application workflows can be programmed in a totally sequential way (no user duties for work partitioning, synchronization and communication).
JS4Cloud functions

*JS4Cloud* implements three additional functionalities, implemented by the set of functions:
- **Data.get**, for accessing one or a collection of datasets stored in the Cloud;
- **Data.define**, for defining new data elements that will be created at runtime as a result of a tool execution;
- **Tool**, to invoke the execution of a software tool available in the Cloud as a service.

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Access</td>
<td>Data.get('&lt;dataName&gt;');</td>
<td>Returns a reference to the data element with the provided name.</td>
</tr>
<tr>
<td></td>
<td>Data.get(new RegExp('&lt;regular expression&gt;'));</td>
<td>Returns an array of references to the data elements whose name match the regular expression.</td>
</tr>
<tr>
<td>Data Definition</td>
<td>Data.define('&lt;dataName&gt;');</td>
<td>Defines a new data element that will be created at runtime.</td>
</tr>
<tr>
<td></td>
<td>Data.define('&lt;arrayName&gt;[&lt;dim1&gt;,...&lt;dimn&gt;]');</td>
<td>Define an array of data elements.</td>
</tr>
<tr>
<td></td>
<td>Data.define('&lt;arrayName&gt;[&lt;dim1&gt;...&lt;dimn&gt;]');</td>
<td>Define a multi-dimensional array of data elements.</td>
</tr>
<tr>
<td>Tool Execution</td>
<td>&lt;toolName&gt;(&lt;par1&gt;,...&lt;parm&gt;);</td>
<td>Invokes an existing tool with associated parameter values.</td>
</tr>
</tbody>
</table>

Script-based applications

- Code-defined workflows are fully equivalent to graphically-defined ones:

```javascript
// Code-defined workflow:
var a = [1, 2, 3];
var b = ['a', 'b', 'c'];
var c = 4.5;
var d = function(x) {
  return x * 2;
};

// Workflow defined graphically:
Workflow = function() {
  var a = createArray([1, 2, 3]);
  var b = createArray(['a', 'b', 'c']);
  var c = createNumber(4.5);
  var d = function(x) {
    return x * 2;
  };
  return new Workflow(a, b, c, d);
}()
```
**JS4Cloud patterns**

**Single task**

```
var DRef = Data.get("Customers");
var nc = 5;
var MRef = Data.define("ClustModel");
K-Means({dataset:DRef, numClusters:nc, model:MRef});
```

![Diagram of Single task process]

**Pipeline**

```
var DRef = Data.get("Census");
var SDRef = Data.define("SCensus");
Sampler({input:DRef, percent:0.25, output:SDRef});
var MRef = Data.define("CensusTree");
J48({dataset:SDRef, confidence:0.1, model:MRef});
```

![Diagram of Pipeline process]
JS4Cloud patterns

Data partitioning

```javascript
var DRef = Data.get("CovType");
var TrRef = Data.define("CovTypeTrain");
var TeRef = Data.define("CovTypeTest");
PartitionerTT({dataset:DRef, percTrain:0.70,
                   trainSet:TrRef, testSet:TeRef});
```

---

JS4Cloud patterns

Data partitioning

```javascript
var DRef = Data.get("NetLog");
var PRef = Data.define("NetLogParts", 16);
Partitioner({dataset:DRef, datasetParts:PRef});
```
JS4Cloud patterns

Data aggregation

```javascript
var M1Ref = Data.get("Model1");
var M2Ref = Data.get("Model2");
var M3Ref = Data.get("Model3");
var BMRef = Data.define("BestModel");
ModelChooser({model1:M1Ref, model2:M2Ref, model3:M3Ref, bestModel:BMRef});
```

---

JS4Cloud patterns

Data aggregation

```javascript
var MsRef = Data.get(new RegExp("^Model"));
var BMRef = Data.define("BestModel");
ModelChooser({models:MsRef, bestModel:BMRef});
```
JS4Cloud patterns

Parameter sweeping

```javascript
var TRef = Data.get("TrainSet");
var nMod = 5;
var MRef = Data.define("Model", nMod);
var min = 0.1;
var max = 0.5;
for(var i=0; i<nMod; i++)
  J48({dataset:TRef, model:MRef[i],
       confidence:(min+i*(max-min)/(nMod-1))});
```

Input sweeping

```javascript
var nMod = 16;
var MRef = Data.define("Model", nMod);
for(var i=0; i<nMod; i++)
  J48({dataset:TsRef[i], model:MRef[i],
       confidence:0.1});
```
Parallelism exploitation

```javascript
var DRef = data.get("OmniRef");
var TRef = Data.define("TrainSet");
var testRef = Data.define("testSet");
var KN - 0.1, nst - 0.5, npmd - 10;
var MRef = Data.define("Model");
var RFRef = Data.define("PredModel");
Partition(trainDRef, percTrain: 0.70, trainSet: TRef, testSet: testRef); for (int i = 0; i < nmd; i++)

ModelSelector(testSet: TRef, model: Model, bestModel: MRef);
```

Monitoring interface

- A snapshot of the application during its execution monitored through the programming interface.
Performance evaluation

- Input dataset: **46 million tuples**
- Used Cloud: **up to 64 virtual servers** (single-core 1.66 GHz CPU, 1.75 GB of memory, and 225 GB of disk)

```
1: var n = 64;
2: var DRef = Data.get("KDDCup99,6GB");
3: TrainRef = Data.define("TrainSet");
4: TestRef = Data.define("TestSet");
5: PartitionersTT([dataset:DRef, percTrain:0.7, trainSet:TrainRef, testSet:TestRef]);
6: var PRef = Data.define("TrainsetPart", n);
7: Partitioner([dataset:TrainRef, datasetPart:PRef]);
8: var MRef = Data.define("Model", n);
9: for(var i=0; i<n; i++)
10: J48([dataset:PRef[i], model:MRef[i], confidence:0.1]);
11: var CRef = Data.define("ClassTestSet", n);
12: for(var i=0; i<n; i++)
13: Classifier([dataset:TrainRef, model:MRef[i], classDataset:CRef[i]]);
14: var FRef = Data.define("FinalClassTestSet");
15: Voter(classData:CRef, finalClassData:FRef);
```

Turnaround and speedup

107 hours (4.5 days)

2 hours
**Efficiency**

![Efficiency Graph]

**Another application example**

- **Ensemble learning workflow**  
  (gene analysis for classifying cancer types)
- **Turnaround time:** 162 minutes on 1 server, 11 minutes on 19 servers.
- **Speedup:** 14.8

```
1. var df = data.get("dataset1");
2. var x = [5, 10, 15, 20, 25, 30, 35, 40, 45, 50];
3. var y = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11];
4. for (var i = 0; i < x.length; i++) {
   var model1 = new Model();
   var model2 = new Model();
   var model3 = new Model();
   // ... predict and evaluate...
}
5. var accuracy1 = model1.getAccuracy();
6. var accuracy2 = model2.getAccuracy();
7. var accuracy3 = model3.getAccuracy();
8. var ensembleAccuracy = (accuracy1 + accuracy2 + accuracy3) / 3;
9. var speedup = (x.length - 1) / 19;
10. data.set("output", ensembleAccuracy, speedup);
```

53

54
Final remarks

- Data mining and knowledge discovery tools are needed to support finding what is interesting and valuable in big data.

- Cloud computing systems can effectively be used as scalable platforms for service-oriented data mining.

- Design and programming tools are needed for simplicity and scalability of complex data analysis processes.

- The DMCF and its programming interfaces support users in implementing and running scalable data mining.

---

Big is quite a moving target?

- Yes, but not only for the increasing size of data.

- Big is a misleading term.

- It must include complexity (and difficulty) of handling huge amounts of heterogeneous data.

- In the first report (2001) on Big Data the term Big was not actually used.
Big is quite a moving target? Some example

- Some data challenges examples we face today
  - Scientific data produced at a rate of hundreds of gigabits-per-second that must be stored, filtered and analyzed.
  - Ten millions of images per day that must be mined (analyzed) in parallel.
  - One billion of tweets/posts queried in real-time on an in-memory database.

Are the challenges faced today different from the challenges faced 10 years ago?

- **Yes**, because data sources are much more than 10 years ago.
- **Yes**, because we want to solve more complex problems.
- **No**, because in data mining we still work on data produced with different goals.
Is Size/Volume the most important issue in Knowledge Discovery?

- **No**, volume is only one dimension of the problem.
- The most important issue is **Value**.
- Size and complexity represent the problem, value is the real benefit.

Smart algorithms and scalable systems

Combination of
- Big data analytics and knowledge-discovery techniques with
- **scalable** computing systems
  for
- an **effective strategy** for producing new insights in a shorter period of time.
- **Clouds** can help.
Data mining for social good

- It’s now time for the public sector to invest in Big data collection and analysis.
- It could improve the quality of life of citizens and the efficiency of public administrations.
- European countries must do more in this area.

Ongoing & future work

- **DtoK Lab** is a startup that originated from our work in this area.

  ![DtoK Lab](www.scalabledataanalytics.com)

- The **DMCF** system is delivered on public clouds as a high-performance Software-as-a-Service (SaaS) to provide innovative data analysis tools and applications.
- Applications in the area of **social data analysis, urban computing, air traffic** and others have been developed by JS4Cloud.
Some publications


Thanks

Questions?

Credits:
Eugenio Cesario, Carmela Comito,
Deborah Falcone, Paolo Trunfio