Programming Script-based Data Analytics Workflows on Clouds

F. Marozzo, Domenico Talia, P. Trunfio

University of Calabria, ICAR-CNR, DtoK Lab
Italy
talia@dimes.unical.it

CAMPUS DI ARCAVACATA

ICAR CNR
Institute for High-Performance Computing and Networking

DtoK Lab
Scalable Data Analytics

Goals

- Using Cloud services for **scalable execution** of **data analysis** applications (expressed as workflows).
- Defining a script-based **programming model** for the **Data Mining Cloud Framework** (**DMCF**).
- Implementing the **JS4Cloud** language (based on that model).
- Evaluating the performance of **JS4Cloud** data mining workflows on **DMCF**.

Talk outline

1. Introduction
2. The Data Mining Cloud Framework
3. The key features of JS4Cloud
   a. Programming concepts
   b. Functions
   c. Patterns
4. Performance figures
5. Final remarks

Introduction

- Workflows have proven to be effective in describing **complex data analysis** tasks and applications linking
  - data sources,
  - filtering tools,
  - mining algorithms,
  - Visualization tools, and
  - knowledge models.
- Data analysis workflows often include **concurrent compute-intensive tasks** that can be efficiently executed on scalable computing infrastructures like Clouds.

Introduction

- We use Cloud services for scalable execution of data analysis workflows in the ***Data Mining Cloud Framework*** (**DMCF**).
- Data analysis workflows in **DMCF** are DAGs originally designed only through a *visual programming interface*.
- Visual programming is an effective design approach for high-level users (domain-expert analysts with limited coding skill).
- In **DMCF task and data parallelism is implicit** (data-driven).

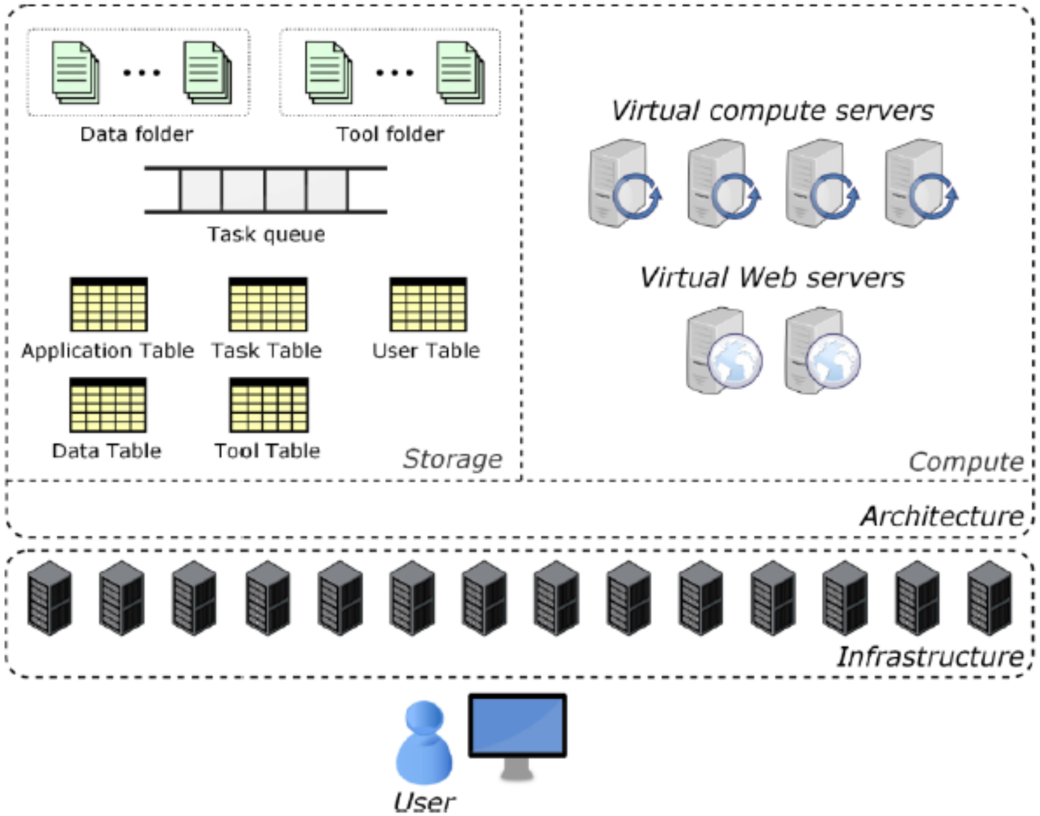
Introduction

- Recently we extended **DMCF** adding a ***script-based data analysis programming model*** as a more flexible programming interface.
- A script language allows experts to program complex applications more rapidly, in a more concise way and with higher flexibility.
- The idea is to provide a script-based data analysis language as an **additional and more flexible programming interface** to skilled users.

Data Mining Cloud Framework

- **Virtual Compute Servers** for executing the workflow tasks.
- **Virtual Web Servers** for the user interface.
- A **Data Folder** of input data and results.
- A **Tool Folder** for libraries and task code.
- **Data Table** and **Tool Table** for metadata of data sources and tools.
- **Application Table**, **Task Table**, and **Users Table**.
- **Task Queue** of tasks ready to be executed.

DMCF

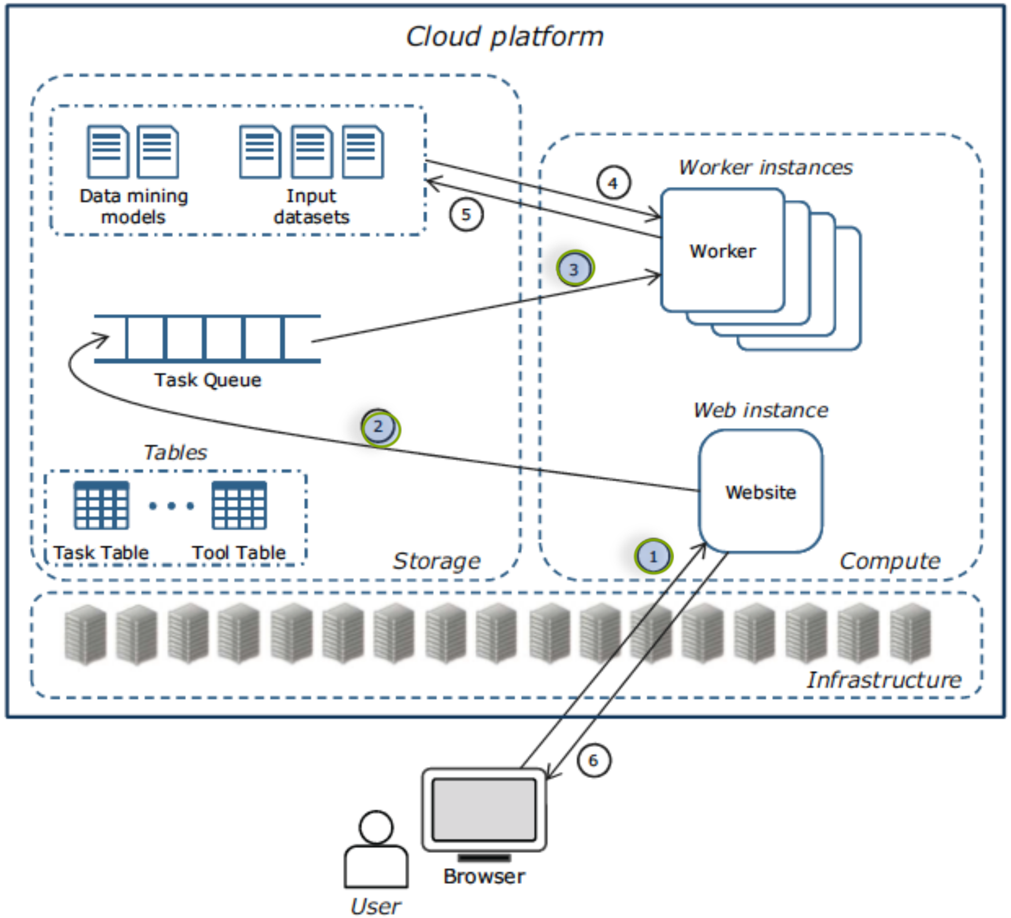


Data Mining Cloud Framework

① Users **access** a Website and **design** the application workflows.

② After submission, the **runtime** selects the workflow tasks and inserts them into the Task Queue on the basis of dependencies.

③ Each idle Virtual compute server picks a task from the Task Queue, and concurrently **forks its execution**.

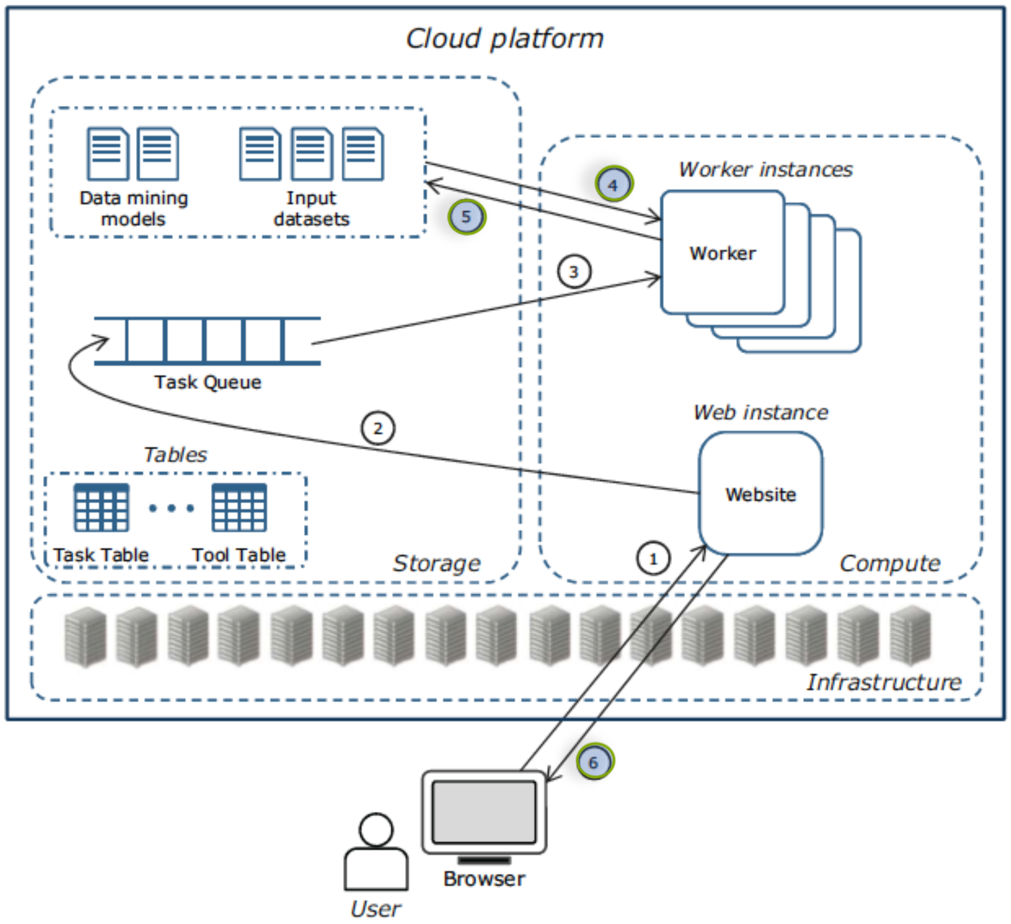


Data Mining Cloud Framework

④ Each Virtual compute server gets the input dataset(s) from its location.

⑤ After task completion, each Virtual compute server stores the **result** in a data storage element.

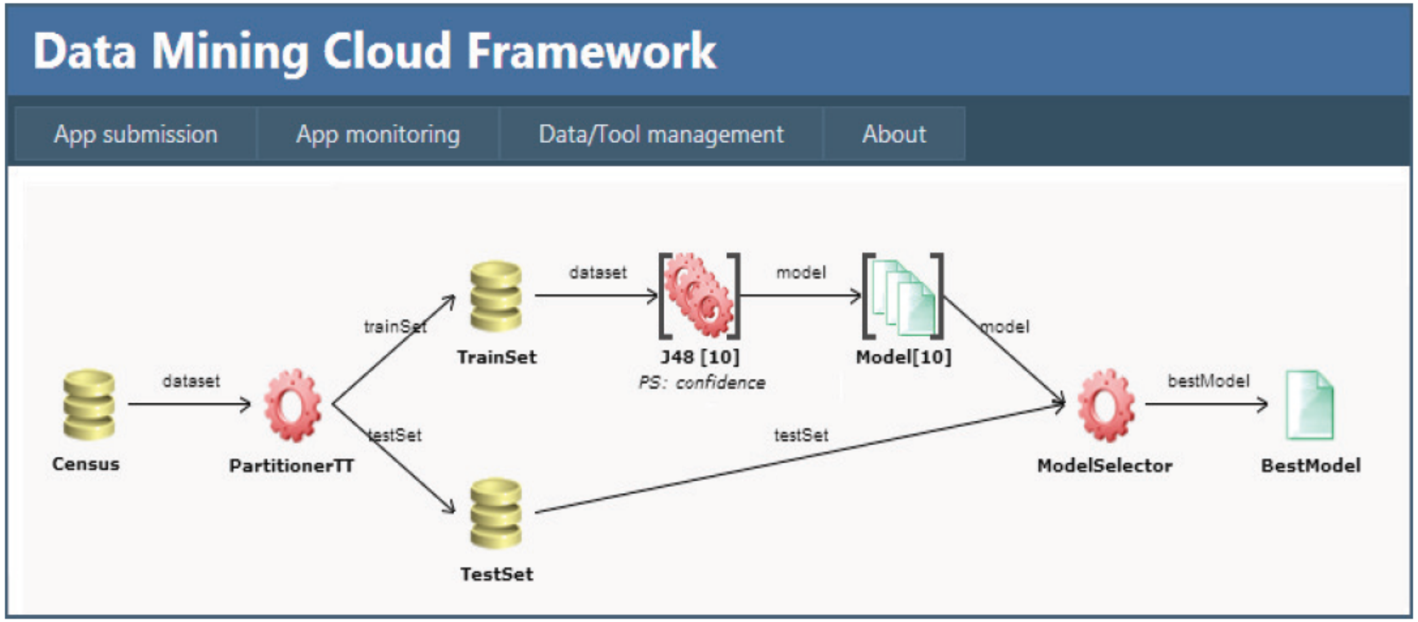
⑥ The Website notifies the user as soon as her/his task(s) have completed, and allows her/him to **access** the **results**.



Visual workflows in DMCF

Workflows includes two types of nodes:

- **Data nodes** represent input or output data elements. A data node can be a ***Dataset*** or a ***Model*** created by data analysis (e.g., a decision tree).
- **Tool nodes** represent tools performing any kind of operation that can be applied to a data node (filtering, splitting, data mining, etc.).



The JS4Cloud script language

- **JS4Cloud** (*JavaScript for Cloud*): 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 parallelism** and **data-driven task parallelism** that automatically spawns ready-to-run tasks to the available Cloud resources;
  - it exploits **implicit task 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:

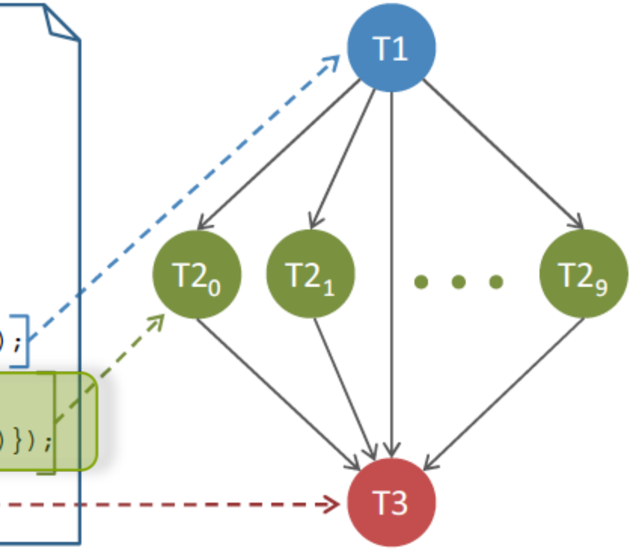
- **Data.get**, for accessing one or a collection of datasets stored in the Cloud;
- **Data.define**, for defining new data elements that will be created at runtime as a result of a tool execution;
- **Tool**, to invoke the execution of a software tool/module available in the Cloud as a service.

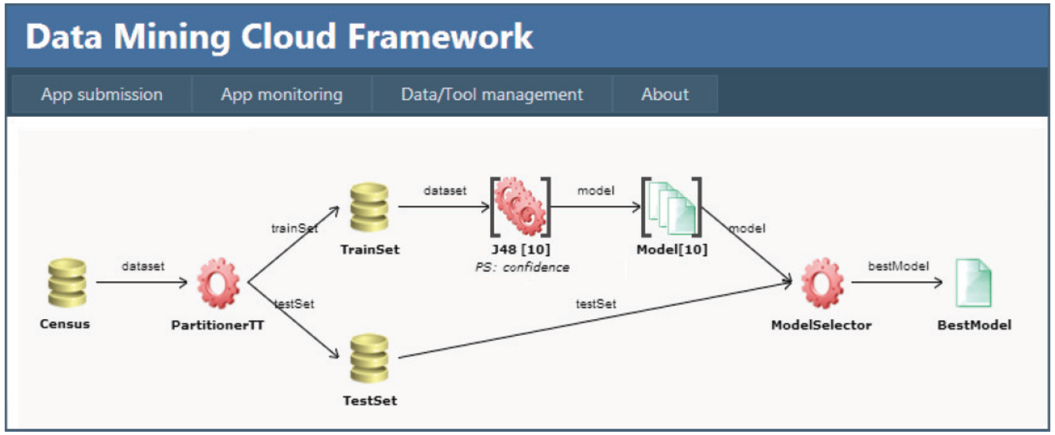
Functionality	*Function*	*Description*
Data Access	Data.get($<\mathit{dataName}>$);	Returns a reference to the data element with the provided name.
	Data.get(new RegExp($<\mathit{regular\ expression}>$));	Returns an array of references to the data elements whose name match the regular expression.
Data Definition	Data.define($<\mathit{dataName}>$);	Defines a new data element that will be created at runtime.
	Data.define($<\mathit{arrayName}>$,$<\mathit{dim}>$);	Define an array of data elements.
	Data.define($<\mathit{arrayName}>$,[$<\mathit{dim}_1>$,...,$<\mathit{dim}_n>$]);	Define a multi-dimensional array of data elements.
Tool Execution	$<\mathit{toolName}>$($<\mathit{par}_1>$:$<\mathit{val}_1>$,...,$<\mathit{par}_n>$:$<\mathit{val}_n>$);	Invokes an existing tool with associated parameter values.

Task Parallelism exploitation

```
var DRef = Data.get("Census");
var TrRef = Data.define("TrainSet");
var TeRef = Data.define("TestSet");
var min = 0.1, max = 0.5; nMod = 10;
var MRef = Data.define("Model", nMod);
var BRef = Data.define("BestModel");

PartitionerTT({dataset:DRef, percTrain:0.70, trainSet:TrRef, testSet:TeRef});
for(int i=0; i<nMod; i++)
   J48({dataset:TrRef, model:Model[i], confidence:(min+i*(max-min)/(nMod-1))});
ModelSelector({testSet:TeRef, model:Model, bestModel:BRef});
```

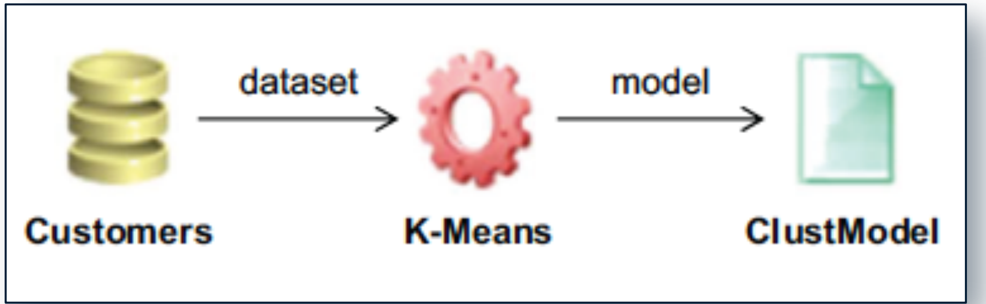




JS4Cloud patterns

Single task

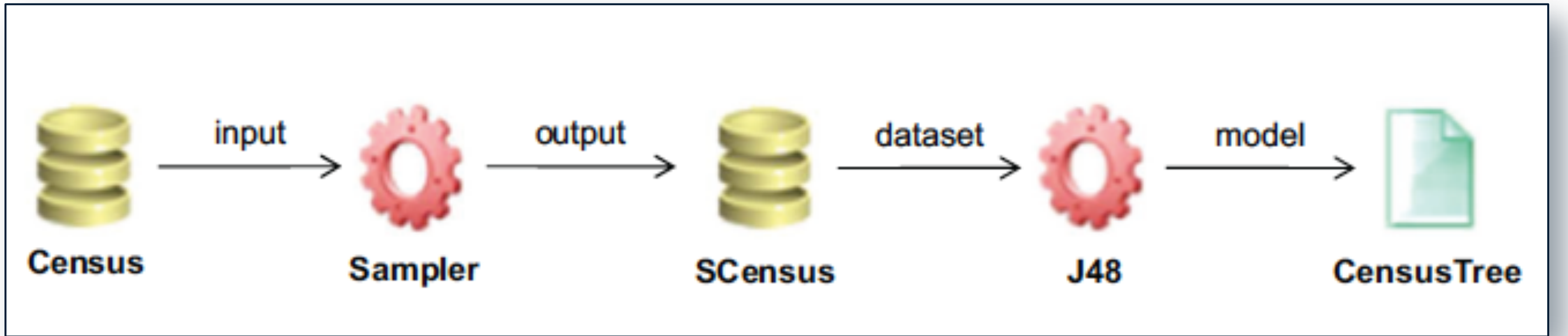
```
var DRef = Data.get("Customers");
var nc = 5;
var MRef = Data.define("ClustModel");
K-Means({dataset:DRef, numClusters:nc, model:MRef});
```



JS4Cloud patterns

Pipeline

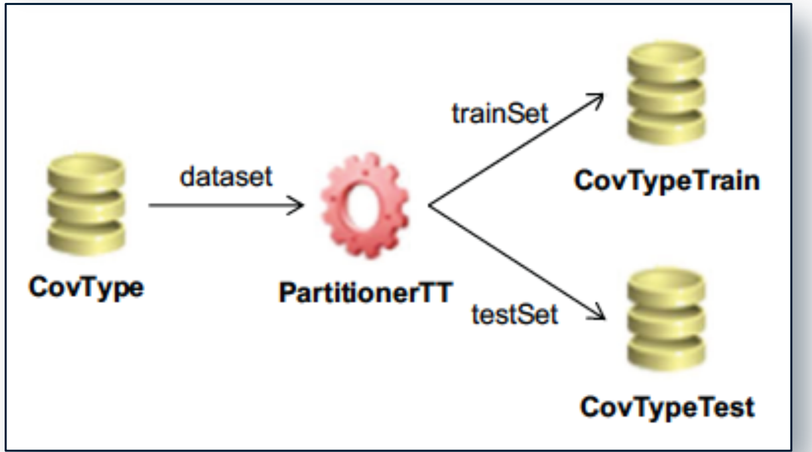
```
var DRef  = Data.get("Census");
var SDRef = Data.define("SCensus");
Sampler({input:DRef, percent:0.25, output:SDRef});
var MRef  = Data.define("CensusTree");
J48({dataset:SDRef, confidence:0.1, model:MRef});
```



JS4Cloud patterns

Data partitioning

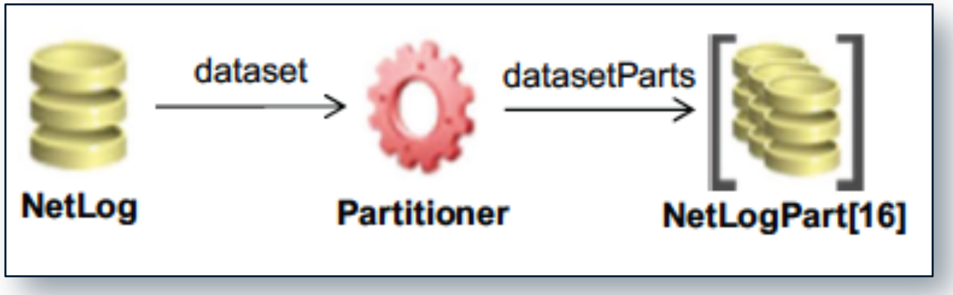
```
var DRef  = Data.get("CovType");
var TrRef = Data.define("CovTypeTrain");
var TeRef = Data.define("CovTypeTest");
PartitionerTT({dataset:DRef, percTrain:0.70,
               trainSet:TrRef, testSet:TeRef});
```



JS4Cloud patterns

Data partitioning (2)

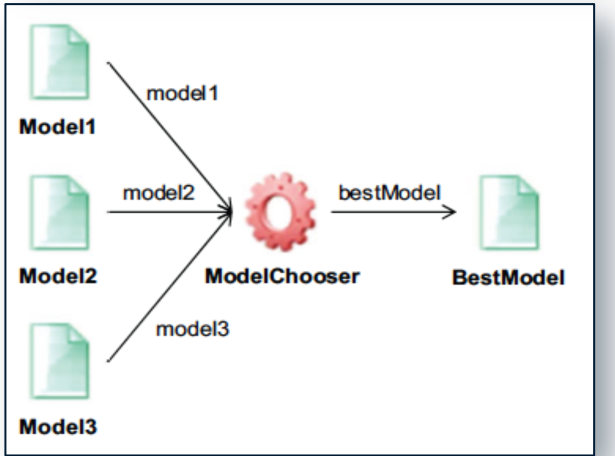
```
var DRef = Data.get("NetLog");
var PRef = Data.define("NetLogParts", 16);
Partitioner({dataset:DRef, datasetParts:PRef});
```



JS4Cloud patterns

Data aggregation

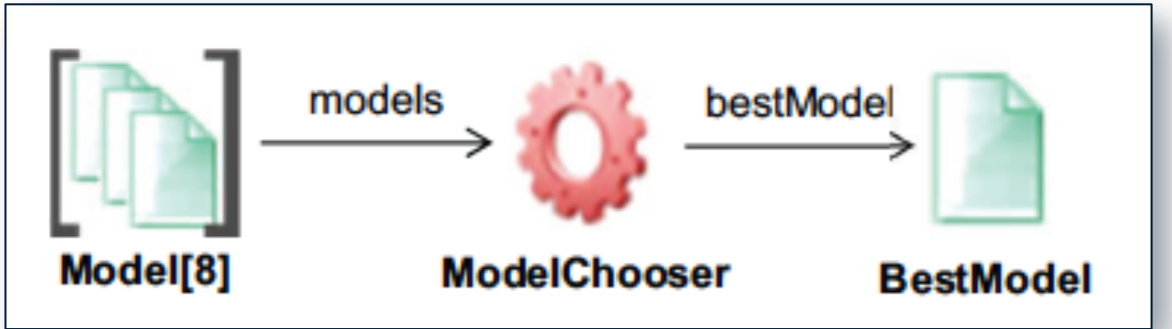
```
var M1Ref = Data.get("Model1");
var M2Ref = Data.get("Model2");
var M3Ref = Data.get("Model3");
var BMRef = Data.define("BestModel");
ModelChooser({model1:M1Ref, model2:M2Ref,
              model3:M3Ref, bestModel:BMRef});
```



JS4Cloud patterns

Data aggregation (2)

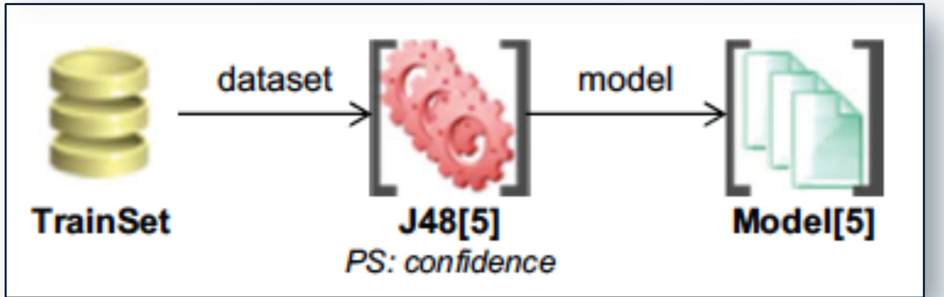
```
var MsRef = Data.get(new RegExp("^Model"));
var BMRef = Data.define("BestModel");
ModelChooser({models:MsRef, bestModel:BMRef});
```



JS4Cloud patterns

Parameter sweeping

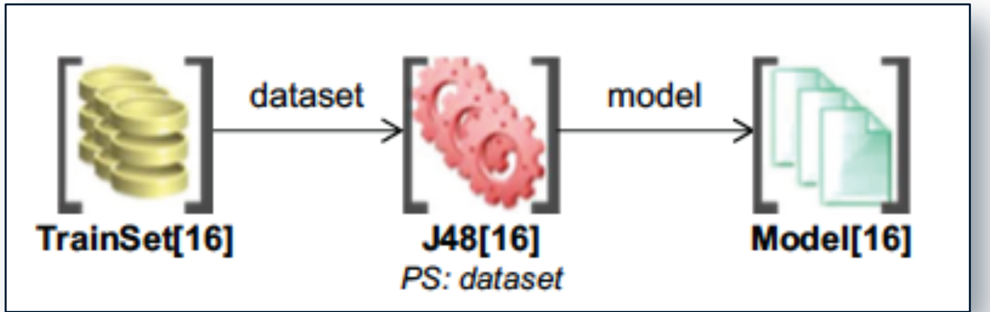
```
var TRef = Data.get("TrainSet");
var nMod = 5;
var MRef = Data.define("Model", nMod);
var min = 0.1;
var max = 0.5;
for(var i=0; i<nMod; i++)
   J48({dataset:TRef, model:MRef[i],
        confidence:(min+i*(max-min)/(nMod-1))});
```



JS4Cloud patterns

Input sweeping

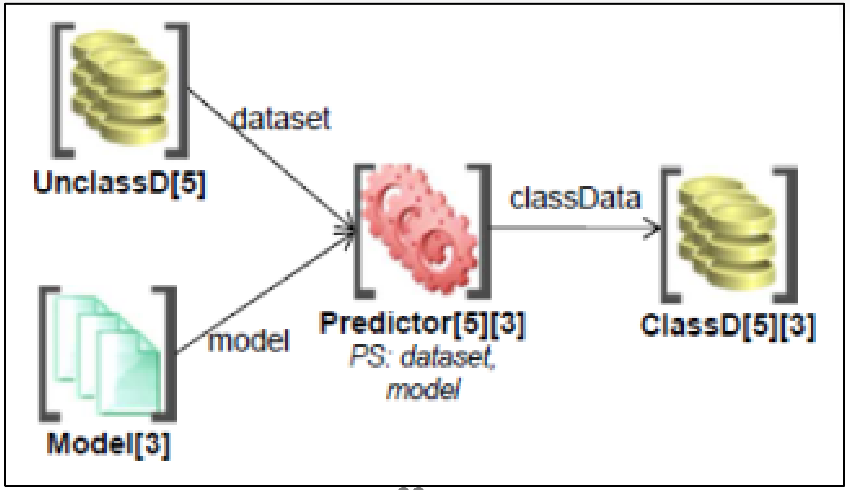
```
var nMod = 16;
var MRef = Data.define("Model", nMod);
for(var i=0; i<nMod; i++)
  J48({dataset:TsRef[i], model:MRef[i], confidence:0.1});
```



JS4Cloud patterns

Input sweeping (2)

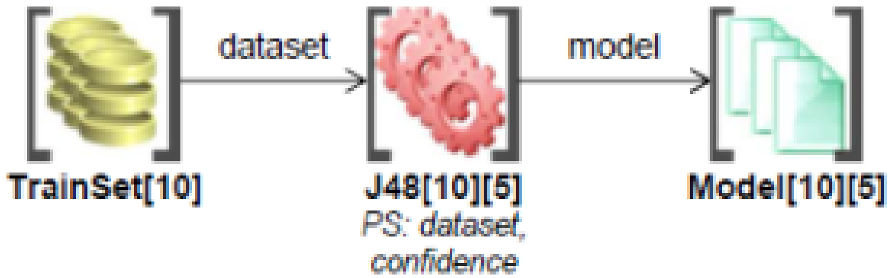
```
var nData = 5, nMod = 3;
var CRef = Data.define("ClassD", [nData, nMod]);
for(var i=0; i<nData; i++)
  for(var j=0; j<nMod; j++)
    Predictor({dataset:DRef[i], model:MRef[j], classDataset:CRef[i][j]});
```



JS4Cloud patterns

Input/Parameter sweeping

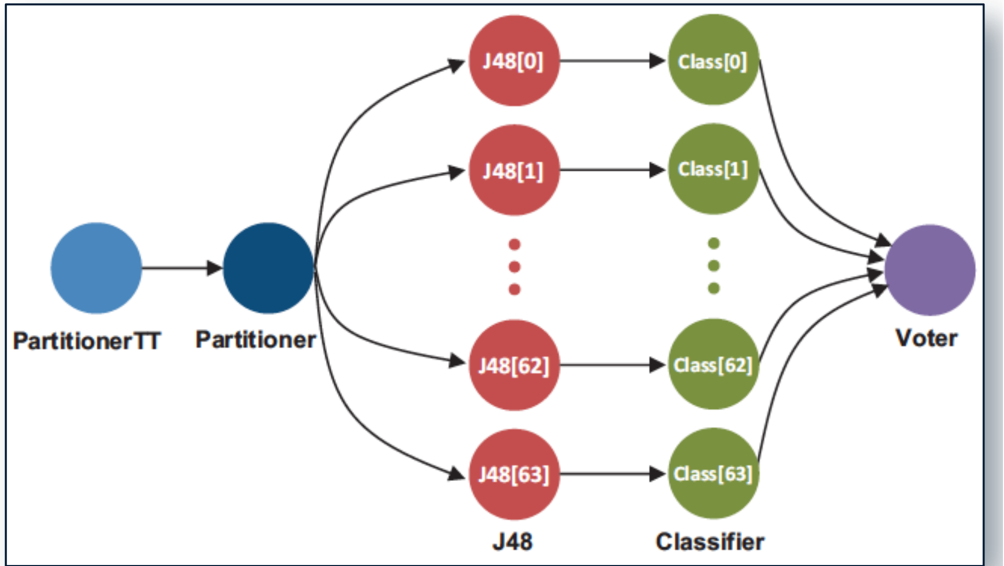
```
var nTr = 10;
var conf = [0.1, 0.2, 0.3, 0.4, 0.5];
var MRef = Data.define("Model", [nTr, conf.length]);
for(var i=0; i<nTr; i++)
    for(var j=0; j<conf.length; j++)
        J48({dataset:TsRef[i], model:MRef[i][j], confidence:conf[j]});
```



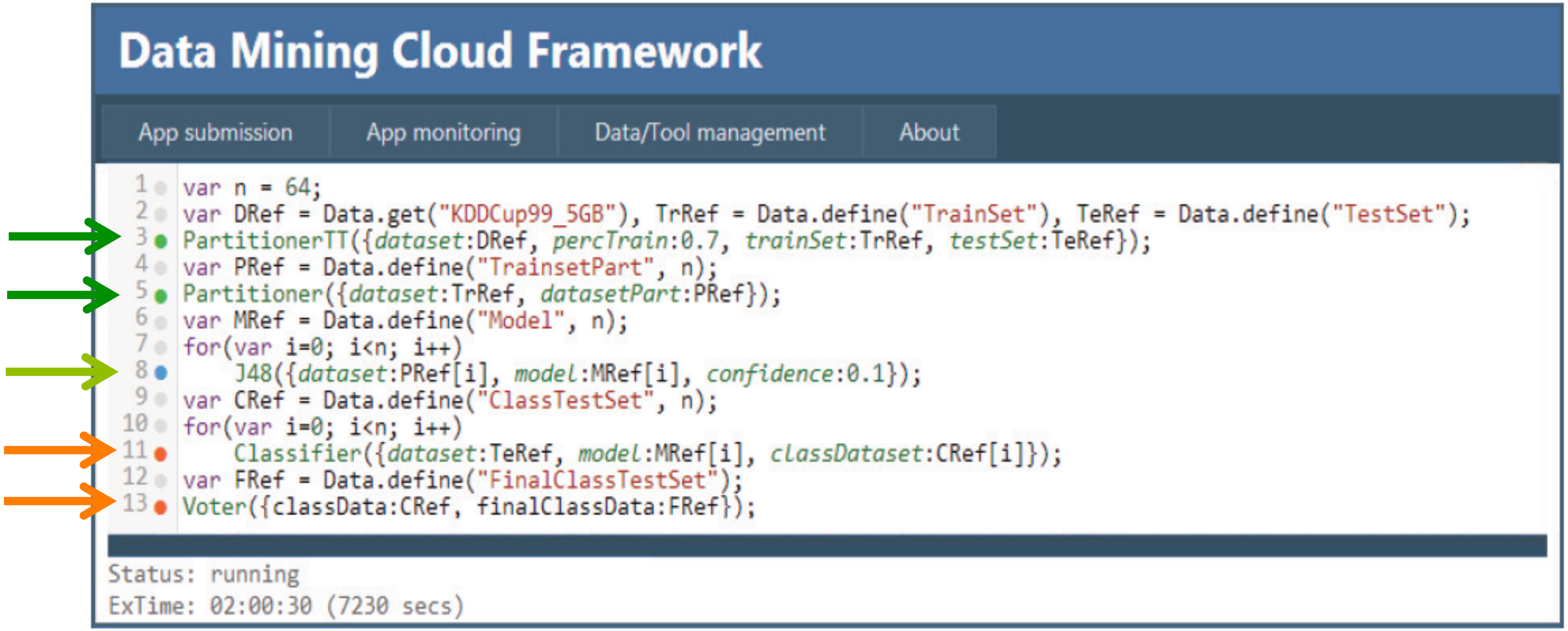
Performance evaluation

- Input dataset: **46 million tuples** (size: 5 GB).
- Used Cloud: **up to 64 virtual servers** (single-core 1.66 GHz CPU, 1.75 GB of memory, and 225 GB of disk)

```
var n = 64;
var DRef = Data.get("KDDCup99_5GB"),
    TrRef = Data.define("TrainSet"),
    TeRef = Data.define("TestSet");
PartitionerTT({dataset:DRef, percTrain:0.7,
    trainSet:TrRef, testSet:TeRef});
var PRef = Data.define("TrainsetPart", n);
Partitioner({dataset:TrRef, datasetPart:PRef});
var MRef = Data.define("Model", n);
for(var i=0; i<n; i++)
   J48({dataset:PRef[i], model:MRef[i],
      confidence:0.1});
var CRef = Data.define("ClassTestSet", n);
for(var i=0; i<n; i++)
   Classifier({dataset:TeRef, model:MRef[i],
      classDataset:CRef[i]});
var FRef = Data.define("FinalClassTestSet");
Voter({classData:CRef, finalClassData:FRef});
```

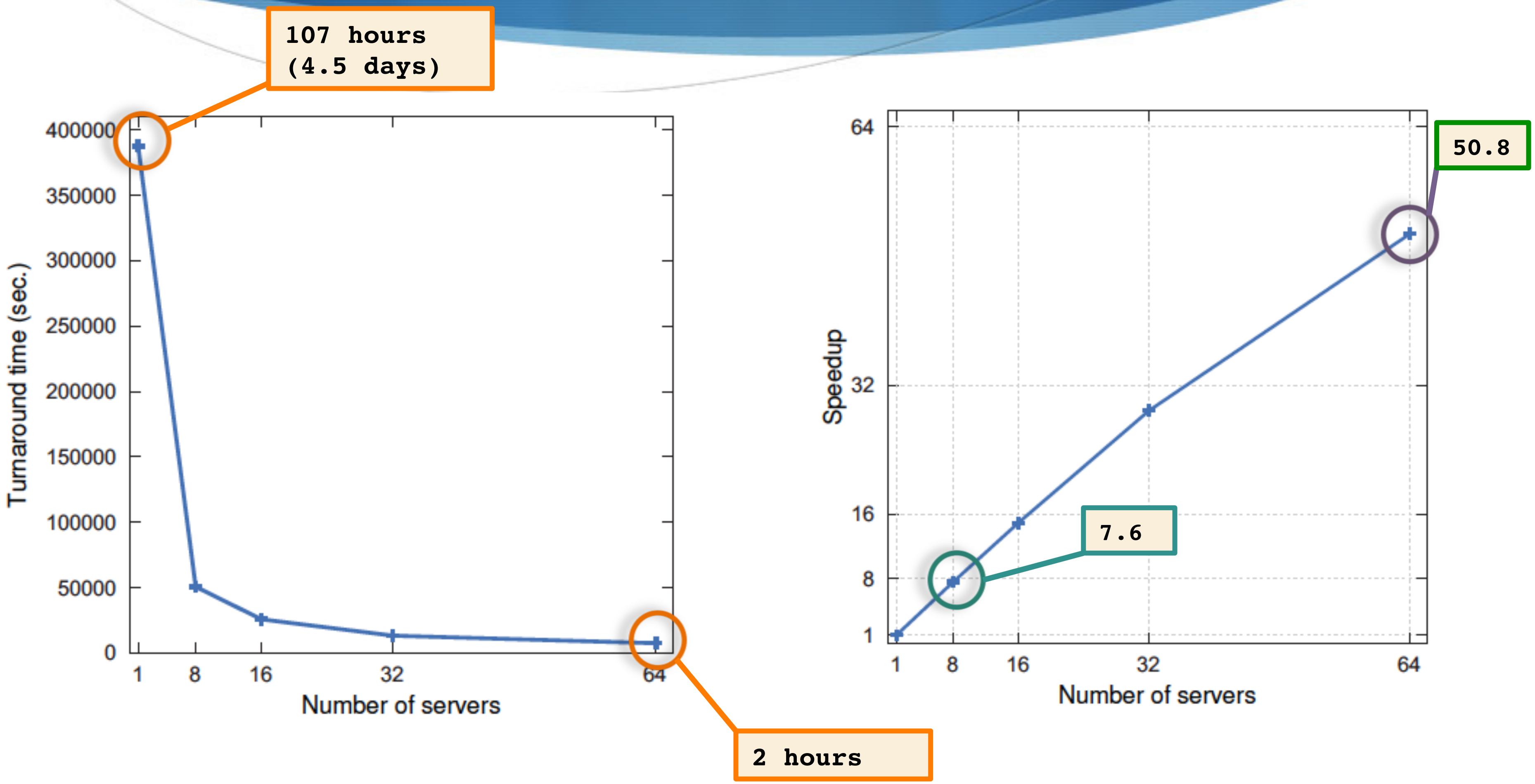


Monitoring interface

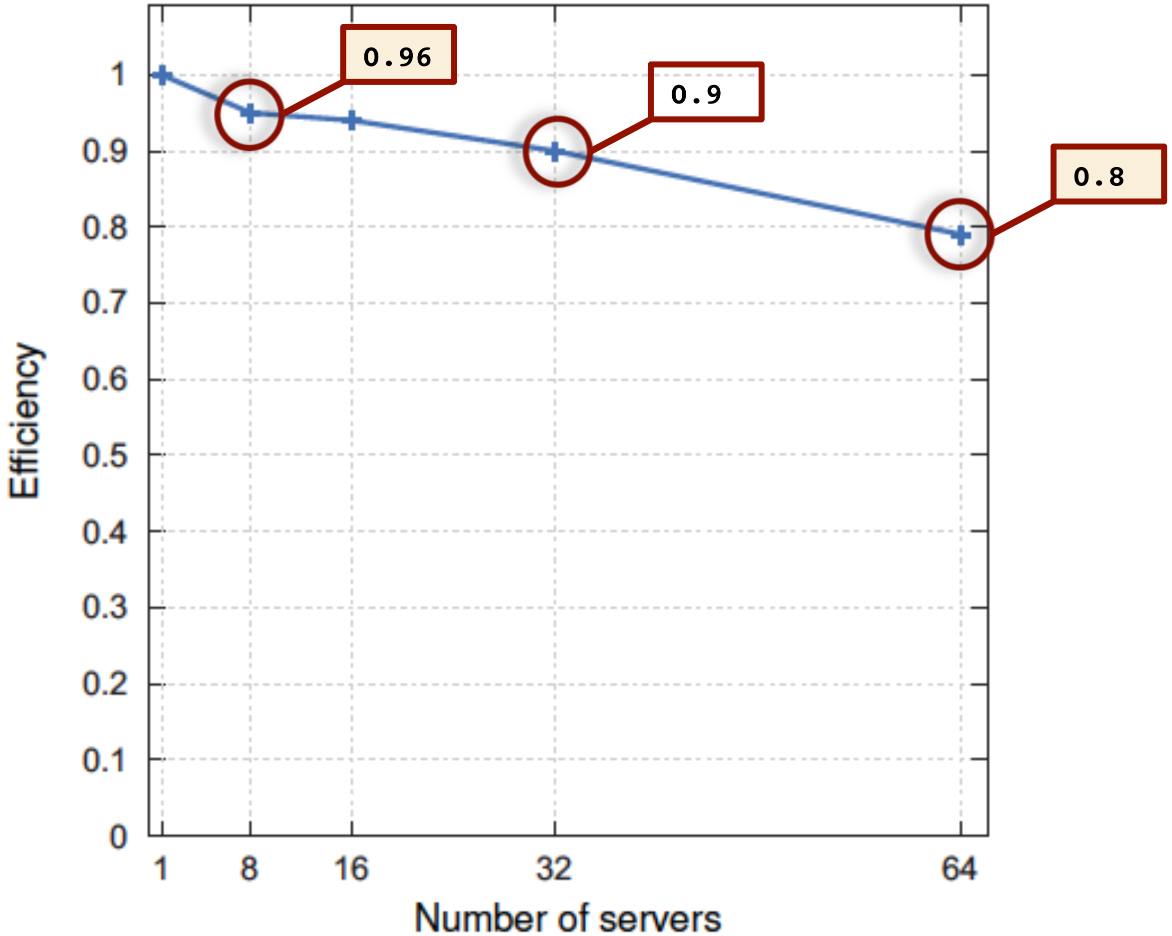


- A snapshot of the application during its execution monitored through the programming interface.

Turnaround and speedup



Efficiency

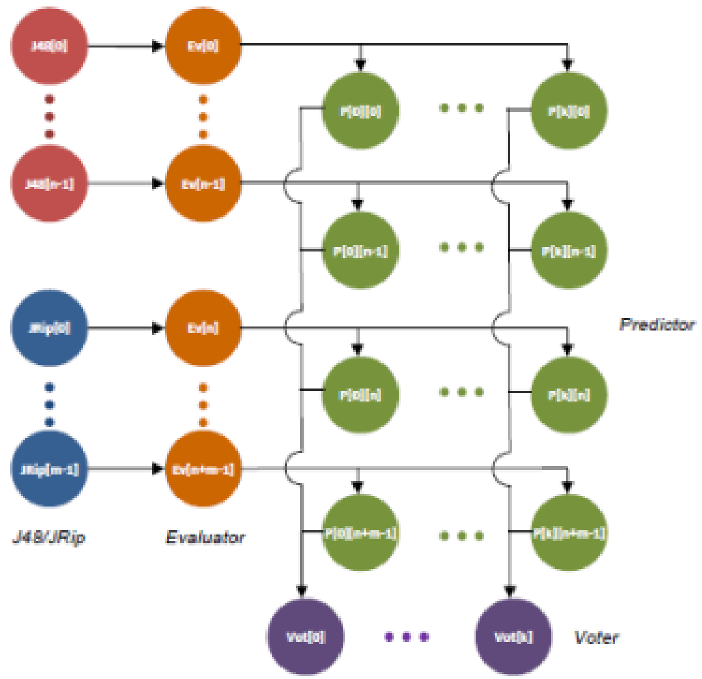


0.96
0.9
0.8
Efficiency
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
8
16
32
64
Number of servers

Another app 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

```
var TrRef = Data.get("GCM-train");
var conf = [0.1, 0.25, 0.5], mno = [2, 5, 10], nfol = [3, 5, 10],
    snum = [1487, 5741, 7699];
var n = conf.length*mno.length, m = nfol.length*snum.length;
var M1Ref = Data.define("Model1", n), M2Ref = Data.define("Model2", m);
for(var i=0; i<conf.length; i++)
   for(var j=0; j<mno.length; j++)
      J48({dataset:TrRef, model:M1Ref[i*mno.length+j], confidence:conf[i],
         minNumObj:mno[j]});
for(var i=0; i<nfol.length; i++)
   for(var j=0; j<snum.length; j++)
      JRip({dataset:TrRef, model:M2Ref[i*snum.length+j], numFolds:nfol[i],
         seed:snum[j]});
var TeRef = Data.get("GCM-test"), EvM1Ref = Data.define("EvModel1", n),
    EvM2Ref = Data.define("EvModel2", m);
for(var i=0; i<n; i++)
   Evaluator({dataset:TeRef, model:M1Ref[i], evalModel:EvM1Ref[i]});
for(var i=0; i<m; i++)
   Evaluator({dataset:TeRef, model:M2Ref[i], evalModel:EvM2Ref[i]});
var k = 4;
var DRef = Data.get("UnlabGCM", k), CRef = Data.define("ClassGCM",[k,n+m]);
for(var i=0; i<k; i++){
   for(var j=0; j<n; j++)
      Predictor({dataset:DRef[i], model:M1Ref[j], classDataset:CRef[i][j]});
   for(var j=0; j<m; j++)
      Predictor({dataset:DRef[i], model:M2Ref[j], classDataset:CRef[i][n+j]});
}
var FRef = Data.define("FinalClassGCM", k), EvMRef = EvM1Ref.concat(EvM2Ref);
for(var i=0; i<k; i++)
   WeightedVoter({classDataset:CRef[i], evalModel:EvMRef, finalClassDataset:FRef[i]});
```



Final remarks

- Main benefits of JS4Cloud are:
  a. it is based on a well known scripting language, so that users do not have to learn a new language from scratch;
  b. it implements a **data-driven task parallelism** that automatically spawns ready-to-run tasks to the available Cloud resources and **data parallelism**;
  c. by exploiting implicit parallelism, application **workflows can be programmed in a totally sequential way**, **users are free from duties like work partitioning, synchronization** and **communication**.

- Experimental performance results prove the effectiveness of the proposed language for programming data analysis workflows
  - **Scalability** can be achieved by executing such workflows on a public Cloud infrastructure.

Ongoing & future work

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

DtoK Lab Scalable Data Analytics 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.

Thanks

COAUTHORS

Fabrizio Marozzo

Paolo Trunfio