FIU-Miner: A Fast, Integrated, and User-Friendly System for Data Mining in Distributed Environment

Chunqiu Zeng, Yexi Jiang, Li Zheng,
Jingxuan Li, Lei Li, Hongtai Li,
Chao Shen, Wubai Zhou, Tao Li
School of Computer Science
Florida International University
{czeng001,yjian004,lzhen001,taoli}
@cs.fiu.edu

Bing Duan, Ming Lei,
Pengnian Wang
ChangHong COC Display Devices Co., Ltd
35 East Mianxing High-Tech Park
Mianyang, Sichuan, China 621000
{bing.duan,thunder,wpn}
@changhong.com

ABSTRACT

The advent of Big Data era drives data analysts from different domains to use data mining techniques for data analysis. However, performing data analysis in a specific domain is not trivial; it often requires complex task configuration, onerous integration of algorithms, and efficient execution in distributed environments. Few efforts have been paid on developing effective tools to facilitate data analysts in conducting complex data analysis tasks.cloud:jplc2011

In this paper, we design and implement FIU-Miner, a Fast, Integrated, and User-friendly system to ease data analysis. FIU-Miner allows users to *rapidly configure a complex data analysis task* without writing a single line of code. It also helps users *conveniently import and integrate different analysis programs.* Further, it significantly *balances resource utilization and task execution in heterogeneous environments.* A case study of a real-world application demonstrates the efficacy and effectiveness of our proposed system.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Application-Data Mining

Keywords

Distributed Environment;Hadoop;Data Mining;Workflow

1. INTRODUCTION

As the data scale and complexity increase explosively, the tasks of discovering knowledge from data have far beyond the processing ability of human being. In many application domains such as healthcare, manufactures, and finance, a typical data mining task often requires complex task configuration, integration of different types of data mining algorithms, and efficient execution on distributed computing environments. Therefore, it is imperative to build tools for data analysts in such domains to efficiently perform data analysis tasks.

Existing data mining products, such as Weka [1], SPSS, and SQL Server Data Tools, provide user-friendly interfaces to facilitate users to conduct the analysis. However, these products are designed for small scale data analysis and do not allow users to plug in new algorithms easily. The data mining algorithm libraries, such as Mahout [5], MLC++ [4], and MILK [3], include a large number of data mining algorithms. However, it requires advanced programming skills to use these libraries for task configuration and algorithm integration in a complex data mining task. Integrated data mining frameworks, such as Radoop [7] and BC-PDM [10], provide user-friendly interfaces to quickly configure data mining tasks. However, these frameworks are Hadoop-based and have limited support for non-Hadoop implementations. Moreover, they do not explicitly address the resource allocation under multi-user and multi-task scenarios.

To address the limitations of existing products, we develop FIU-Miner to facilitate data analysts to efficiently perform data mining tasks. FIU-Miner provides a set of novel functionalities that help data analysts conveniently and efficiently conduct complex data mining tasks. Specifically, the system has the following significant merits:

- *User-friendly rapid data mining task configuration.* Following the Software-as-a-Service paradigm, FIU-Miner hides the low-level details irrelevant to the data analysis tasks. Through the interface, users can easily configure a complex data mining task by assembling existing algorithms into a workflow without writing a single line of code.
- *Flexible cross-language program integration.* FIU-Miner is able to make full use of the existing state-of-the-art data mining tools by allowing users to import them into its system algorithm library. There is no restriction on the choice of implementation languages for the imported programs, since FIU-Miner is capable of correctly distributing the tasks to appropriate computing nodes with suitable runtime environments.
- *Effective resource management in heterogeneous environments.* FIU-Miner supports the data mining tasks running in heterogeneous environments, including graphics workstations, stand-alone computers, and computing clusters. To optimize the resource utilization of the available computing resources, FIU-Miner schedules the tasks by con-

KDD'13, August 11–14, 2013, Chicago, Illinois, USA.

sidering various factors such as algorithm implementation, server load balance, and the data location.

2. SYSTEM OVERVIEW

FIU-Miner has been designed and developed by a research team consisting of 12 members for one year. A demo can be found in http://datamining-node08.cs.fiu.edu/FIU-Miner. Figure 1 presents an overview of the system architecture of FIU-Miner. The system is divided into four layers: *User Interface*, *Task and System Management*, *Abstracted Resource*, and *Heterogeneous Physical Resource*.

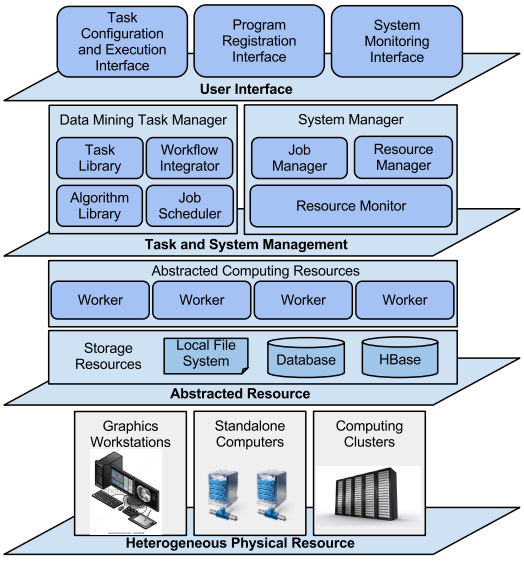

Figure 1: System Architecture.

2.1 User Interface Layer

To maximize the system compatibility, the user interface is presented as a pure HTML5 web-based application. There are three major modules of user interfaces and their functionalities and features are described as follows:

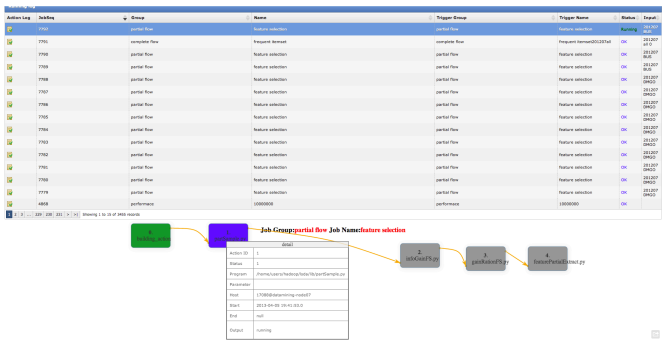

Figure 2: Data Mining Task Configuration.

- *Task configuration and execution. (Figure 2)* This module supports workflow-oriented task configuration. The workflow of a data mining task is represented and visualized as a directed graph, where nodes denote specific algorithm and edges denote the data dependency among algorithms. A workflow can be quickly configured through GUI instead of programming. Moreover, the users can set the execution plan of a data mining task that can be automatically executed whenever the time is up.

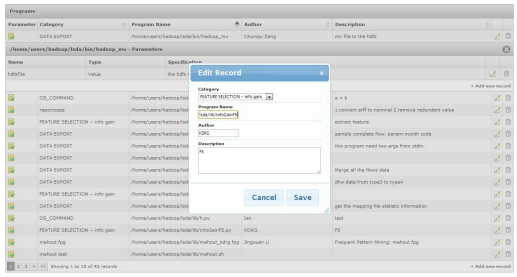

Figure 3: Data Mining Program Registration.

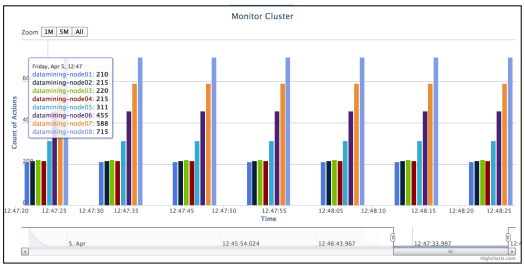

Figure 4: Resource Monitoring.

- *Program registration. (Figure 3)* This module allows users to easily import external data mining programs to enrich the *Algorithm Library*. To import a program, a user needs to upload the executable files and provide the detailed profiles of the program, including functionality description, required runtime environment, dependencies, and parameter specification. The imported program can be written in any form as long as its runtime environment is supported by the backend servers. FIU-Miner currently supports Java (including Hadoop environment), Shell, Python, C/C++, etc. Therefore, almost all the mainstream data mining algorithm implementations, such as Weka-based programs, Mahout-based programs, and MILK-based programs, can be imported into FIU-Miner. Users can also implement and import their own algorithms into the system.
- *System monitoring. (Figure 4)* This module monitors the resource utilization of FIU-Miner in real time, and also tracks the running status of the submitted tasks. Note that this module only displays the abstracted resources (e.g., the available Workers, the tree structures of the file system) for users. The underlying physical resources are transparent to users.

2.2 Task and System Management Layer

This layer contains two major modules: *task management* and *system management*.

2.2.1 Task management

As previously mentioned, users are allowed to configure ad-hoc data mining tasks to fulfill their analysis requirements. A user can pick the available algorithms registered in the *Algorithm Library* as a building block when configuring a data mining task. The *Workflow Integrator* conducts task validation and reports invalid integration. Once

the configuration is done, the new data mining task will be automatically added to the *Task Library* and is ready for scheduling.

The *Job Scheduler* is responsible for assigning the computing jobs to the computing *Worker(s)* where the runtime environments are supported with the purpose of minimizing the running time. The scheduling in FIU-Miner is not trivial as it supports programs of different languages in a heterogeneous environment. It is possible that the programs in a configured job have different runtime requirements; hence, simply assigning the job to an arbitrary worker may render the job inexecutable. On the other hand, the I/O cost would increase when decomposing the job into different steps and running each step to a different worker. The scheduling would become even more difficult when considering the multi-user and multi-task situation. In FIU-Miner, to address the aforementioned challenge, we implement the scheduler by considering the following factors: (1) runtime environment requirements of each step in a given job; (2) supported runtime environment of each computing worker; (3) current running status of each computing worker; and (4) estimated data size of the input.

2.2.2 System management

The *Job Manager* keeps track of the running status of a executed job. A user will be immediately notified the job status in real time.

Besides job monitoring, FIU-Miner also keeps track of the status of the workers and the underling computing resources. The *Resource Monitor* monitors the *Worker*s and provides the running status of the *Workers* to the *Job Scheduler* to help make scheduling decisions. The *Resource Manager* manages all the available *Worker*s. A unique feature of FIU-Miner is that it does not need the manual registration of available physical resources. Once a worker is deployed on a physical server, the worker will automatically register the server to FIU-Miner by sending server profile to the *Resource Manager*.

2.3 Resource Abstraction Layer

In FIU-Miner, the computing power of the physical servers is quantified by the number of *Worker*s and the data mining tasks are scheduled to the *Worker*s. This mechanism, as a simplified version of system virtualization, is able to maximize the utilization of the computing resources.

To effectively manage the computing resources, each worker is associated with a profile containing the detailed specification of its capability, including the computing power, supported runtime environment, and running status, etc. The storage of a physical server is shared by all the tenant *Worker*s, including the available database, HDFS, and local file systems.

3. A CASE STUDY

FIU-Miner has been deployed as the manufacturing process analysis platform at ChangHong Corporation, one of the world's largest display product manufacturing companies located in China. This system is currently running in a distributed environment containing one 64-node computing cluster and several graphics workstations. Each node of the cluster consists of 4 Intel Xeon E5645 CPU, 32GB main memory and 1T hard disk space. And a graphics workstation consists of 1 Intel i7 3820 3.6GHz CPU, 32GB main memory, and NVIDIA Tesla K20 with 3.52T FLOPS computing power.

To demonstrate the effectiveness of FIU-Miner, we present a case study of optimizing the Plasma Display Panel (PDP) manufacturing process using FIU-Miner. PDP manufacturing is a complex process, whose yield ratio highly depends on the parameter setting values associated with each production equipment. Due to the complexity of manufacturing procedure (75 assembling processes with over 300 production equipments), a large number of features (9364 involved parameters), and huge amount of operational data (300G per month), automatic analysis tools are needed for optimizing the production procedure.

One common task in the PDP process optimization is to extract important feature combinations that are highly related to the yield ratio. To accomplish this task, a workflow and its associated components are depicted in Figure 5. The workflow invovles the following steps:

1. PDP dataset is loaded from HDFS by *HDFS Data Loader*, and *Data Publisher* dispatches the dataset to three different *Feature Selection* algorithms including *mRMR* [6], *InfoGain* [8], *ReliefF* [2].
2. Important features are first extracted by the *Feature Selection* algorithms, and then combined by the *Stable Feature Selection* component to output the stable features which are highly ranked by all the feature selection algorithms [9].
3. Based on the selected stable features, the *Finding Frequent Feature Combinations* component is running iteratively until the number of frequent feature combinations reaches the predefined threshold K.
4. The discovered top-K frequent feature combinations are stored into the database.

Generally, three major phases (Component Preparation, Task Configuration and Task Execution) are needed to compose this complex task from the scratch. Table 1 compares the data analysis with and without FIU-Miner and demonstrates the advantages of FIU-Miner.

During Phase I, without FIU-Miner, users can use existing tools at each component; however, they have to implement the data loading and storing procedures between components. By contrast, FIU-Miner provides customized user interfaces for data loading and storing. Users can easily prepare each component and integrate multiple algorithms without writing a single line of code.

During Phase II, without FIU-Miner, users need to manage the dependence among different components. As the number of components increases, their dependence would become more complicated. It is difficult for users to schedule the tasks in a distributed environment. On the contrary, FIU-Miner provides a user-friendly GUI to build the dependence among all the components. It can also take full advantage of distributed environments to improve the efficiency of task execution, and the scheduling is transparent to users.

During Phase III, without FIU-Miner, users need to implement the monitor to track the mining process. There is no efficient mechanism for users to locate the failure components. Comparatively, FIU-Miner provides automatic functionalities to track the running status of each component in the workflow. In addition, users can easily find out the details of the failure components.

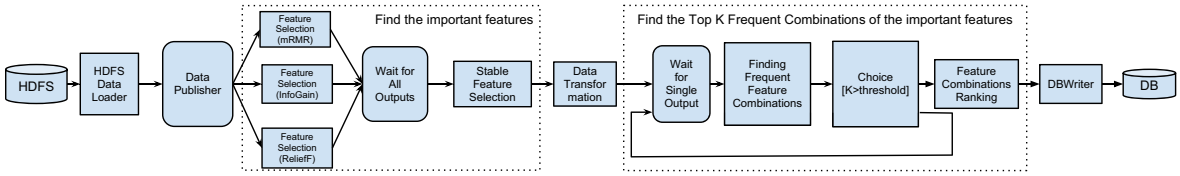

Figure 5: The Workflow for PDP Manufacturing Case Study.

		Phase I: Component Preparation	Phase II: Task Configuration	Phase III: Task Execution
Without FIU-Miner	Tasks	Implement the data loading and storing procedures.	Writing code to manage component dependence; Manual scheduling.	Implement the monitor module; Difficult to locate failure.
	Costs	A few hours for coding.	Up to days.	Up to days.
	Require data analysts to have advanced programming skills with experience in distributed environments.			
With FIU-Miner	Tasks	Rapid data loading and storing using customized interfaces.	Manage dependence using GUI; Scheduling transparent to users.	Monitor and track mining process with user-friendly GUI; Support failure diagnosis.
	Costs	A few clicks.	Within one hour without programming.	A few clicks without programming.
	No programming requirements for data analysts.			

Table 1: PDP Data Analysis using FIU-Miner.

4. RESOURCE UTILIZATION EVALUATION

In this section, we design two sets of experiments to investigate how FIU-Miner utilizes static and dynamic computing environments. To avoid conflicts with the production cluster, we use another 8-node cluster as a separate experiment environment.

Exp I: Each node in the cluster is deployed with one *Worker*. We configure 10 tasks with different running times in FIU-Miner. To mimic the random job submission and task execution, we schedule these jobs to start at time 0 and repeat with a random interval ($<$ 1 minutes). The system is observed in a 80-minutes period and the running status of the system is recorded. Figure 6 shows how FIU-Miner balances the workloads for the underlying infrastructures, where the x-axis denotes the time and the y-axis denotes the average number of completed jobs for each *Worker*. As is shown, the accumulated number of completed jobs (the blue bars) increases linearly, whereas the amortized number of completed jobs (the white bars) remains stable. These phenomena show that FIU-Miner achieves a good balance of the resource utilization by properly distributing jobs.

Figure 6: Load Balance in Static Environment.

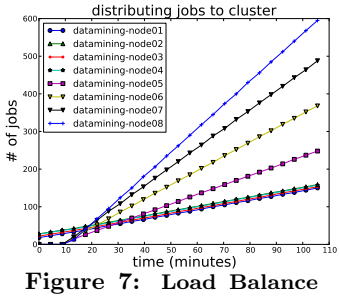

Figure 7: Load Balance in Dynamic Environment.

Exp II: We use the same job setting as in *Exp I*. To investigate the resource utilization of FIU-Miner under a dynamic environment, we initially provide four nodes, each with 1 *Worker*, and then add the other four nodes 10 minutes later. To emulate the nodes with different computing powers, the newly-added nodes are deployed with 2 to 5 *Workers*, respectively. Each *Worker* is restricted to use only 1 CPU core at a time, so the node deployed with more *Workers* can have more powerful computing ability. We observe the system for 110 minutes and Figure 7 shows the number of jobs completed by each node over time. As is shown, the first four nodes with 1 *Worker* have approximately the same number of completed jobs at each observation timestamp. The slopes of the newly-added nodes are higher than those of the first four nodes, since they have more computing powers. We also observe that the number of completed jobs is proportional to the number of *Workers* on each node. This experiment illustrates that FIU-Miner can balance the workloads under a dynamic environment.

5. CONCLUSIONS

In this demo paper, we present FIU-Miner, an integrated system that facilitates users to conduct the ad-hoc data mining tasks. FIU-Miner provides a user-friendly GUI to allow users to rapidly configure complex data mining tasks. It also allows users to conveniently import and integrate arbitrary data mining programs. Internally, FIU-Miner leverages *Job Scheduler* to effectively schedule the data mining jobs and leverages *Resource Manager* to manage the underlying resource management. The case study shows that FIU-Miner is effective in configuring complex task, integrating various algorithms and executing tasks in distributed environments.

6. ACKNOWLEDGEMENT

This project is supported partially by NSF under Grant HRD-0833093 and CNS-1126619, and by the Army Research Office under Grant W911NF-12-1-0431.

7. REFERENCES

[1] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explorations*, 2009.

[2] H. Liu and H. Motoda. *Computational Methods of Feature Selection*. Chapman & Hall, 2008.

[3] MILK. http://pythonhosted.org/milk.

[4] MLC++. http://www.sgi.com/tech/mlc.

[5] Sean Owen, Robin Anil, Ted Dunning, and Ellen Friedman. *Mahout in Action*. Manning, 2011.

[6] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. *IEEE PAMI*, 2005.

[7] Zoltan Prekopcsak, Gabor Makrai, Tamas Henk, and Csaba Gaspar-Papanek. Radoop: Analyzing big data with rapidminer and hadoop. In *RCOMM*, 2011.

[8] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to data mining*. Pearson Education, 2006.

[9] Adam Woznica, Phone Neuyen, and Alexandros Kalousis. Model mining for robust feature selection. In *KDD*, 2009.

[10] Le Yu, Jian Zheng, Bin Wu, Bai Wang, Chongwei Shen, Long Qian, and Renbo Zhang. Bc-pdm: Data mining, social network analysis and text mining system based on cloud computing. In *KDD*, 2012.