PERSONALIZATION AND DATA RELATION EXPLORATION USING PREDICTIVE ANALYTICS FOR THE PRODUCTION AND DISTRIBUTED ANALYSIS SYSTEM (PANDA)

by

MIKHAIL TITOV

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy at
The University of Texas at Arlington
August, 2016

Arlington, Texas

Supervising Committee:

Gergely Záruba, Supervising Professor
Kaushik De
Manfred Huber
David Levine

ABSTRACT

Personalization and Data Relation
Exploration using Predictive Analytics for
the Production and Distributed Analysis System (PanDA)

Mikhail Titov, Ph.D.
The University of Texas at Arlington, 2016

Supervising Professor: Gergely Záruba

Efficient data distribution among computing centers is one of the biggest challenges in large-scale scientific distributed computing systems. Such data distribution issues include: i) the rational utilization of storage and computing resources, ii) the minimization of the completion time for data processing (which requires a reduction in redundant data transfers, and intelligent allocation of processing tasks), and iii) user experience enhancement, i.e., availability and fast access to the desired data, and discovery of new relevant data. In the literature and in practice, there have been significant new approaches to the improvement of workflow management to address the above described issues, especially the first two. However, scientific computing systems usually miss out on enhancing user experience, although significant improvements could be done by exploring the relationships between the involved entities, e.g., inter-user, user-data relationships. Such revealed relationships would not only be to the benefit of the users, but could also improve data distribution strategies.

The focus of this dissertation is on the discovery of hidden correlations between users and corresponding data, and on the interpretation of the reasons of those correlations in terms of a quantitative assessment.

The scientific computing system on which this research is focused is the pilot-job based workload management system called PanDA (Production and Distributed Analysis) that operates at the ATLAS experiment. The dissertation describes a research effort that was conducted to detect data usage patterns in PanDA to validate a thesis that a recommender system would enhance user experience as well as provide important data with which scheduling of computing tasks could be improved. Data mining techniques are investigated and applied to estimate the correlation between users' data needs, and to collect and manage groupings of data (based on data origin and usage patterns) and users (based on interests and data usage history).

This work also presents the design of Data Watcher, a system that can create and maintain user models and thus reveal relationships between users and data. The goal is to be able to analyze, model, and predict user preferences based on estimated ratings and user provided feedback. The core analytics of Data Watcher is based on various recommender system techniques to provide methods in assisting users in finding interesting data (i.e., data similar to what the user has used previously, or relevant data that similar users have used). More precisely, Data Watcher i) can predict the degree of users' potential interest in particular data, ii) dynamically forms groups of similar objects (groups of similar users, and data collections), and iii) maintains data popularity metrics based on implicit and explicit ratings.

ACKNOWLEDGEMENTS

hun. "*Mindent lehet, csak akarni kell.*"

I would like to express my sincere appreciation to everyone who supported me. I would like to especially thank particular people who positively influenced my professional career, and more specifically, my Ph.D. study.

My sincerest thanks go to my supervising professor Dr. Gergely Záruba, who expertly guided me through my graduate education and provided tremendous support. His enthusiasm and confidence in the success of my research kept me motivated and encouraged. I would like to thank my co-adviser Dr. Kaushik De for his significant impact on my career, and particularly for giving me this opportunity. Furthermore, his intelligence and kindness inspired me to succeed. I am thankful to Prof. David Levine for his support and generosity, and to Dr. Manfred Huber for his passion in everything he does (especially in CS). My deep gratitude goes to Dr. Alexei Klimentov for guiding me during my professional career as my mentor.

I would also like to thank Dr. Yuriy Aleksandrovich Chernyshev and Dr. Nikolay Petrovich Vasilyev for their substantial support at the beginning of my career.

My appreciation also extends to all my colleagues from the Distributed Computing group of the ATLAS experiment, CERN.

August 15, 2016

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Scientific computing has undergone a major transformation in the last decades. Most large science experiments require vast computing and data storage resources in order to provide results or predictions based on the raw data obtained. For scientific distributed computing systems with hundreds of petabytes of data and thousands of users, it is important to keep track not just of how data is distributed in the system (i.e., the efficiency of data distribution among computing centers), but also of individual user's interests in the distributed data. Furthermore, with limited resources for storage it is important that data is replicated only at centers where it will be most likely (and most frequently) accessed. Such information can be discovered based on the variations of user interests in the distributed data. This however requires the collection and use of specific statistics such as correlations between data distributions, user preferences and the mechanics of data distribution and utilization.

This dissertation is in the context of the Production and Distributed Analysis system (PanDA) designed for the use in the ATLAS experiment (one of the largest physics experiments of our time) housed at CERN. The reader is referred to Section 1.2: *Background* for more information about CERN and ATLAS, and Section 1.3: *Related Work* for PanDA description.

1.1 Motivation

This research highlights that correlation between objects in highly dynamic (variable) scientific computing environments (with high rate of the data production

and changes of user interests during data processing campaigns) can be detected and used to extend the capabilities of workload management systems thus increasing their usability. Distributed data management and workload management systems should thus focus on improving the quality of such relationship estimations in order to achieve more efficient resources allocation. Furthermore, the revealed relationships between users, and users and data could help to minimize the computational time spent in such time-sensitive processes as search requests, queuing of computational tasks, retrieving results, etc.

The deep understanding of inter-data relationships (with their corresponding strength) between objects can i) facilitate the rationalization of resource usage, ii) simplify workflow management, while maintaining the quality of service provided. Such understanding could also help introduce novel, and better management strategies and help review scheduling priorities.

1.1.1 Problem Statement

The inherent complexity in the data and workload management systems of scientific distributed computing systems can pose strong limitations on how data and data workflow need to be organized. Most of the current implementations thus have to cope with such limitations resulting in reduced performance and overall the usage experience. As a result, the following negative effects could surface: i) data distribution (i.e., creating additional replicas of popular data) reacts slowly to changes in user interests; ii) it can become difficult to discover new data that is similar to data that was used earlier (as an offline option, it is possible to use shared bulletin boards with the descriptions on what properties the new data contains); iii) it can be burdensome to keep track of data that was used by users of different groups who work on the same analysis tasks.

The current production and analysis ecosystem does not consider user preferences and is thus not well suited to dynamically react to changing user demands, i.e., to predict which data will be popular for certain groups of users and prepare the processing resources ahead of time (i.e., create additional data replicas at particular computing nodes for better computing task scheduling). Data distribution currently does not explicitly follow user interests, and thus distribution of new data does not meet user needs sufficiently and in a timely manner. This also means that redundant data transfers cause an unwanted increase in the waiting time during data processing. These reasons underline the need for a properly designed system that models user interests and uses this model to follow user activity. Such system would be able:

- to predict the degree of user interest in any data,
- to reveal similarity between users and between data, and
- to consider explicit user ratings per data (i.e., the capability to indicate the significance of various data).

This system would also serve as an artificial assistant that could:

- guide users in the discovery of new data that is in the users' interest area (based on an estimated degree of interest in particular data properties),
- help to discover similar users (i.e., users with similar areas of interest),
- inform users about the availability of particular data that comes with personal recommendation,
- provide feedback about any data, i.e., numerical rating to express the significance and the quality of a particular data for the user.

Furthermore, the system provided quantitative estimations of the relationships between user and data will:

- further enhance the quality of distribution of data among computing centers (choosing the appropriate storage type, and the number data replicas),
- improve the quality of computational tasks scheduling (i.e., pre-allocation of computational tasks), and finally
- enrich user experience (i.e., extend user possibilities in management of analysis process).

There are three main classes that need to be distinguished within PanDA:

- data (i.e., data unit),
- location (i.e., storage and/or computing nodes/sites),
- user (i.e., consumer of the processed data according to the provided computing task description).

In terms of distributing data (among storage nodes) or computational tasks (among computing nodes) a *user* could also be considered as one of the many input parameters (as the recipient and the owner) and not strictly as a distinct class to build relations with. Our emphasis is on exploring relations between *user* and *data* classes to be able to extensively utilize this obtained knowledge.

Every user activity in the past (e.g., the number of *connections* between user and data during a particular period of time) form a base for *relation representation* with a corresponding degree that denotes how strong the connection is. Descriptions of user and data objects provide a set of properties that are used to refine the corresponding relation. Relationships between user and data objects are used to build relations inside each class as well, and thus to reveal the degree of similarity between objects of the same class.

The goals for new models of users and relations (i.e., user profile and user activity respectively) are to meet appropriate requirements (of the highly variable environment), i.e.,:

- to keep track of new data that are produced with high rate and distribution of that data might be correlated with user;
- to consider the short lifetime of user interest to a particular data (or data feature) due to data processing campaigns (in the ATLAS experiment, data gets reprocessed relatively frequently, when a better software model is found);
- to keep popularity metrics up to date based on discovered correlations between the data needs of users.

1.1.2 Overview of Methodology

The general approach we have chosen to address the research problem is heavily influenced by the involved entities (user and data), the expected outcomes, and that the actions of users should be able to affect the outcome (e.g., users should be able to provide explicit feedback as well as users' reactions to the system outcomes should trigger implicit feedback mechanisms). Applied analysis should be focused on the revealing of hidden relations between objects and on the discovery of behavioral patterns. Based on the above considerations, data mining provides appropriate techniques in the exploration of data and in the endeavour of finding the most applicable structured information within. During our research, the following classes of tasks were used for solving specific problems:

- association analysis with association rule mining,
- sequential pattern mining,
- classification, and clustering.

Discovered relations and patterns are ranked to express their significance as an input to the decision making process; thus the techniques broadly described as of *information filtering* were applied to build a recommender system as the core of the solution.

At a high level, the main intention of a general recommender system is to provide relevant and valuable recommendations to users (i.e., discover data, that would benefit users); however, the core components in achieving this are data mining processes, which produce the predicted value of the user's interest (likeliness score) in particular data (which is later used when filtering the actual data that is presented to the user). The input sources to the corresponding mining processes are user activity, data description (that could be taken from a system where analysis has already been applied to the data to retrieve such information, or from external information systems), and possibly user feedback as a reaction to the system-presented estimated ratings (i.e., the output of the recommender system may trigger a user to provide a corresponding feedback either as an action, e.g., "accept" or "reject", or as an explicit rating of a particular data or data collection, e.g., 3 out of 5 stars).

1.1.3 Rationale and Significance

Data distribution processes always depend on such factors as: availability (which is usually restricted by storage space), accessibility (time to access, which also depends on the number of data replicas), and efficiency (minimizing time costs, e.g., reducing the number of data transfers between computing centers). To be able to react to temporal variances in users' interest areas, thus to minimize resources when following new user interests, a system that is responsible for data distribution should be aware of relationships between users and data, and thus preferably be able

to predict the popularity of certain data among certain group of users. This can be achieved by revealing usage patterns and correlations between users' data needs.

The understanding of the nature of relations between users and data helps in learning the reasons of user interest in particular data, and predict the degree of interest to the new data (i.e., data that was not used by the same user earlier). More precisely, the comprehensive analysis of user behavior will require a complex description of user needs, but in turn will help to assist users in discovering new data that might be in the user's interest area or would benefit the user.

1.2 Background

This section provides an overview of the environment for the research conducted in this dissertation, that:

- introduces experiments in high energy physics (with an emphasis on the ATLAS experiment) that deal with exploring and analyzing the data produced by the particle detectors;
- describes the computing infrastructure for data distribution used in the ATLAS experiment; and
- presents the ATLAS Computing Model with description of data types and formats, and corresponding policies for data processing and allocation.

1.2.1 The ATLAS Experiment

CERN is the European Organization for Nuclear Research (french "Conseil Européen pour la Recherche Nucléaire"). Its main area of research is particle physics the study of the fundamental constituents of matter and the forces acting between them. The instruments used at CERN are purpose-built particle accelerators and detectors. Accelerators boost beams of particles to high energies before the beams

are made to collide with each other or with stationary targets. Detectors located around possible collision points along the ring observe and record the results of these collisions [1]. The main particle accelerator at CERN and in the world is currently the Large Hadron Collider (LHC).

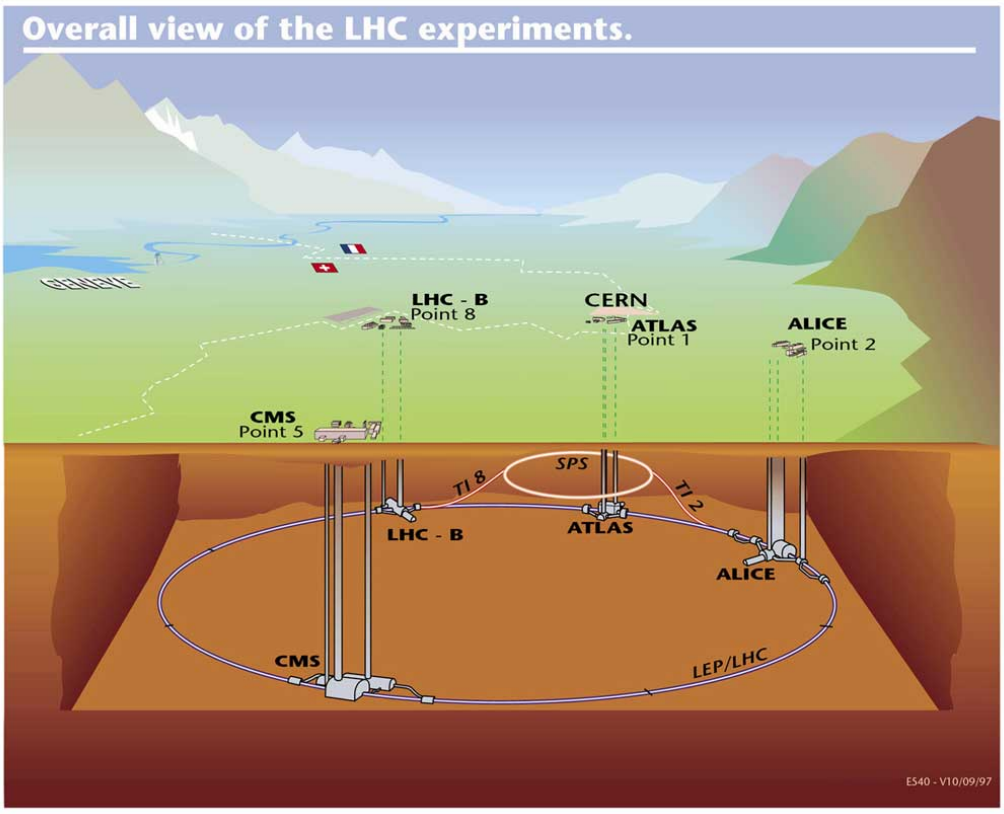


Figure 1.1: The LHC particle accelerator with four main detectors (ALICE, ATLAS, CMS, and LHCb)

The LHC consists of a 27-kilometer long ring surrounded by superconducting magnets with a number of accelerating structures to boost the energy of the particle beams along the way [2]. Inside the accelerator, two high-energy particle beams

[1] http://home.cern/about
[2] http://home.cern/topics/large-hadron-collider

travel in opposite directions close to the speed of light before they are made to collide. Seven experiments at the LHC use detectors to analyse the myriad of particles produced by collisions in the accelerator [3]: ATLAS (A Toroidal LHC ApparatuS), CMS (Compact Muon Solenoid), ALICE (A Large Ion Collider Experiment), LHCb (Large Hadron Collider beauty), TOTEM (TOTal Elastic and diffractive cross section Measurement), LHCf (Large Hadron Collider forward), and MoEDAL (Monopole and Exotics Detector at the LHC). Figure 1.1 [4] illustrates the LHC particle accelerator with the four main detectors.

The context of this dissertation is the ATLAS experiment, which investigates a wide range of physics, from the search for the Higgs boson to extra dimensions and particles that could make up dark matter. Although it has the same scientific goals as the CMS experiment, it uses different technical solutions and a different magnet-system design.

The ATLAS detector (Figure 1.2 [5]) is nominally forward-backward symmetric with respect to the interaction point. The magnet configuration comprises a thin superconducting solenoid surrounding the inner-detector cavity, and three large superconducting toroids (one barrel and two end-caps) arranged with an eight-fold azimuthal symmetry around the calorimeters. This fundamental choice has driven the design of the rest of the detector [1].

Beams of particles from the LHC collide at the center of the ATLAS detector making collision debris in the form of new particles, which fly out from the collision point in all directions. Six different detecting subsystems arranged in layers around the collision point record the paths, momentum, and energy of the particles, allowing

[3] http://home.cern/about/experiments
[4] http://atlasexperiment.org/photos/lhc.html
[5] http://atlasexperiment.org/photos/full-detector-cgi.html

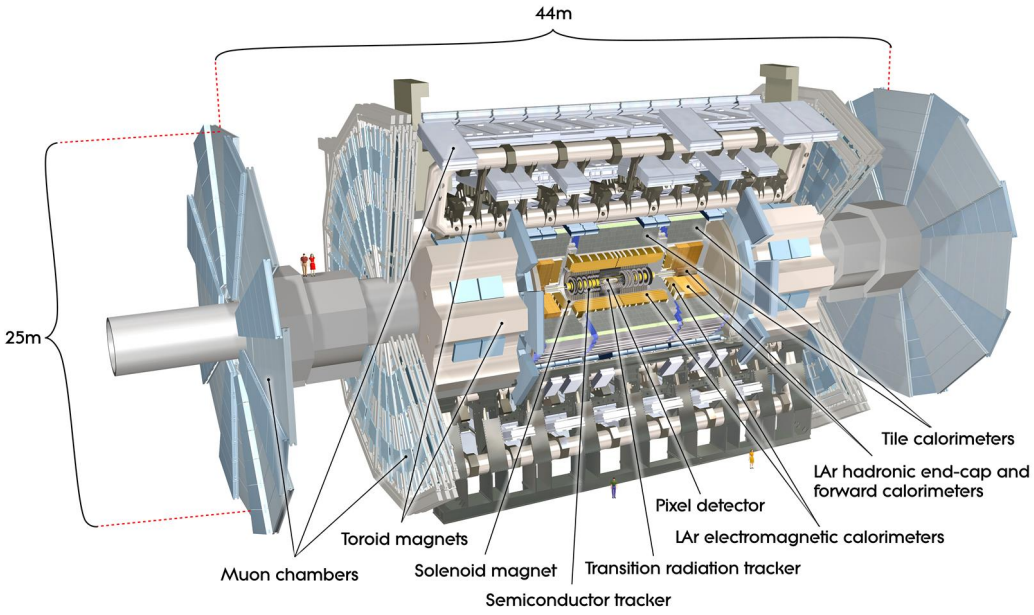


Figure 1.2: A detailed computer-generated image of the ATLAS detector and its systems

them to be individually identified. A huge magnet system bends the paths of charged particles so that their momenta can be measured.

Over a billion particle interactions take place in the ATLAS detector every second, and only one in a million collisions are flagged as potentially interesting and recorded for further study. *Events* are the lowest significant units of data in high energy physics, representing a collision captured in the detector.

1.2.2 The Worldwide LHC Computing Grid

Hundreds of research institutions participate in analysis of data from the LHC, and are connected by a distributed computing infrastructure called the Worldwide LHC Computing Grid (WLCG). Distributed computing resources for analysis by end-user physicists are provided by the European Grid Infrastructure (EGI/EGEE),

the Open Science Grid (OSG), and the Nordic Data Grid Facility (NDGF) with NorduGrid middleware (ARC, Advanced Resource Connector).

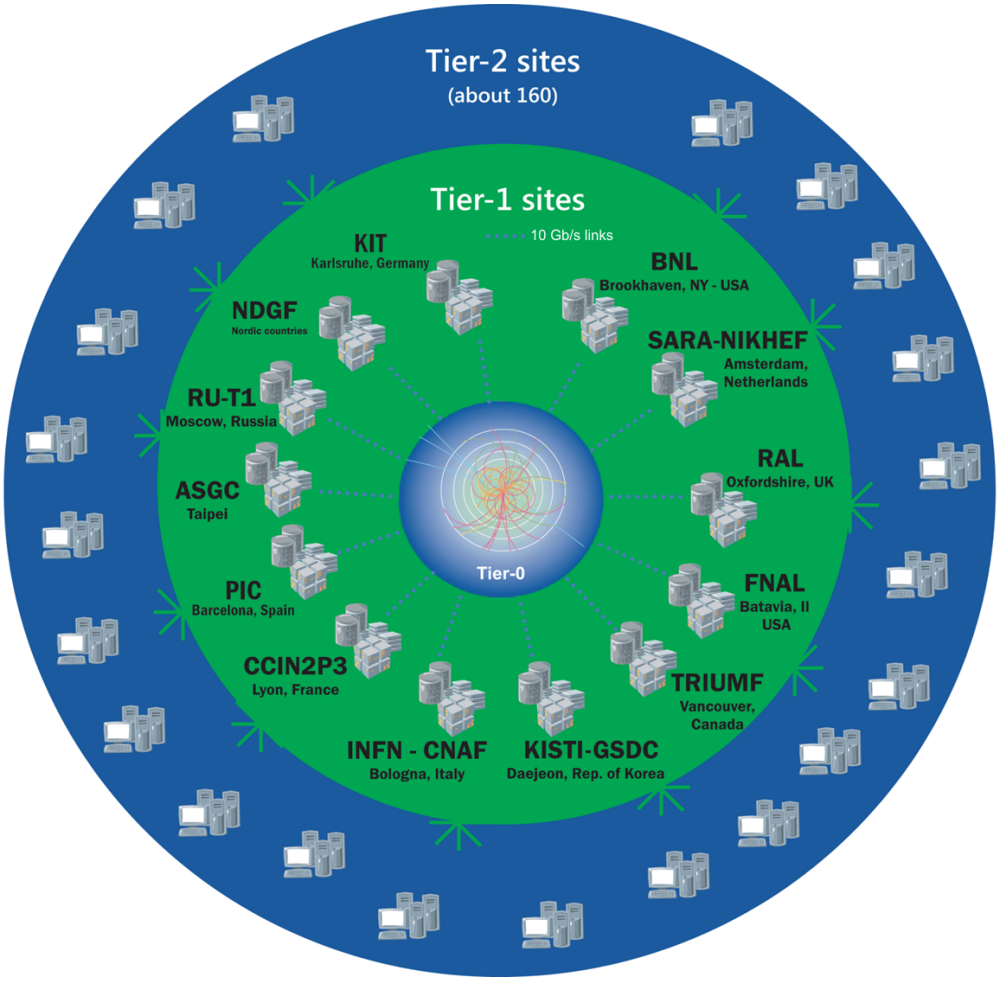


Figure 1.3: WLCG Tier centers

The WLCG is composed of four levels, or "Tiers", numbered from 0 to 3 (see Figure 1.3 [6]). Each Tier is made up of several computer centers and provides a specific set of services. Tier-0 is the CERN Data Center. It is responsible for the safe keeping of the raw data and performs the first pass at reconstructing the raw data

[6] http://wlcg-public.web.cern.ch/tier-centres

into meaningful information. Tier-0 distributes data to Tier-1s. There are 13 Tier-1 designated computer centers. They are responsible for storing a proportion of raw and reconstructed data, as well as performing large-scale reprocessing; distributing data to Tier-2s; and storing a share of the simulated data that the Tier-2s produce. Tier-2s are typically hosted by universities and other scientific institutes that can store sufficient data and provide adequate computing power for specific analysis tasks. Individual scientists can access the Grid through local (or Tier-3) computing resources.

The structure of the WLCG can be described with the four main component layers: networking, hardware, middleware, and physics analysis software. Exchanging data between WLCG centers is managed by the Grid File Transfer Service (FTS). Each Tier's grid center manages a large collection of computers and storage systems with specialized storage tools, such as dCache system, CERN Advanced STORage system (CASTOR), etc. Middleware represents the software infrastructure which allows access to distributed computing resources and archives, and is able to support complicated and time-consuming data analysis (thus connects operating systems of the computers with the physics applications software). The immense and changing demands of the high energy physics environment require dedicated software to analyse vast amounts of data efficiently. One of the main physics analysis software is ROOT [7].

Recent updates to the grid infrastructure loosen the strictly hierarchically structured roles of the Tier centers to be able to best use their full set of capabilities (e.g., use Tier-2s for reconstruction of MC [8] data), but keeps certain levels of requirements (e.g., the quality of service) the same [2].

[7] ROOT is an object-oriented data analysis framework based on C++, specifically designed for large scale data analysis.

[8] Simulated Monte Carlo data (during ATLAS data taking MC data are produced with the same releases as are used for online selection and offline reconstruction of the real data).

1.2.3 The ATLAS Computing Model

The ATLAS Computing Model embraces the "grid paradigm" and calls for a high degree of decentralisation and sharing of computing resources [3]. Its goal is to form a model for a production and analysis ecosystem which would provide seamless access to all ATLAS data and resources [4]. Thus, a complex set of tools and distributed services are provided, enabling the automatic distribution and processing of the large amounts of data. This infrastructure consists of the following two main building blocks:

- the Athena software framework, with its associated modular structure of the event data model, including the software for: event simulation, event trigger, event reconstruction, and physics analysis tools; and
- the Distributed Computing tools that are built on top of Grid middleware: the Distributed Data Management (DDM) system, the Distributed Production system, the Ganga/pAthena frameworks for distributed analysis on the Grid, monitoring and accounting.

1.2.3.1 Data Representation

Data organization is represented with different level of granularities for book-keeping: *events* (that are collected during an operational period of the detector or generated by simulation mimicking various physics process) are stored in *files*, and files with events of the same experimental property are grouped into *datasets*. Thus, a dataset can be (loosely) defined as a collection (i.e., aggregation) of files (that are processed together and usually comprise the input or output of a computation or data acquisition process) plus associated metadata. Furthermore, every file can be part of multiple datasets at the same time when it is involved in various data management

activities. Knowing that a dataset represents files that are used together, the system can optimise its units of data transfer and discovery.

1.2.3.2 The Event Data Model

The physics event store holds a number of successively derived event representations, beginning with raw or simulated data and progressing through reconstruction into more streamlined event representations suitable for analysis [3]. Corresponding Event Data Model defines a number of different data formats:

- RAW data are events as output by the Event Filter in "byte-stream" format, reflecting the format in which data are delivered from the detector, rather than in any object-oriented representation. Each file contains events belonging to a single run (corresponding to a prolonged period of data taking using the same trigger selections on the same fill in the accelerator), but the events in each file will not be consecutive nor ordered.
- ESD (Event Summary Data) is event data written as the output of the reconstruction process (including low level information as hits/tracks and cells/clusters), produced from the RAW data; it has an object-oriented format (POOL [9]/ROOT).
- AOD (Analysis Object Data) is a reduced event representation (i.e., summary of event reconstruction with "physics" objects), derived from ESD, suitable for analysis; it has an object-oriented format (ROOT). xAOD is a new AOD format that is completely redesigned for analysis use, thus is readable by both ROOT (for high level reconstruction objects such as jets or muon tracks), and Athena (allowing full access to all objects) [2].

[9]POOL is a hybrid technology store for C++ objects, using a mixture of streaming and relational technologies to implement both object persistency and object metadata catalogs and collections (Persistency Framework).

- DPD (Derived Physics Data) is a representation for end-user analysis, that is produced for working groups or individual end-users (group-specific format). Derived data dESD (for performance groups) and dAOD (for physics groups) are produced from ESD and AOD respectively, and are aimed to reduce the overall dataset size and therefore improve the processing time by either skim events (event selection) or slim (reduce the information of selected objects), trim (removing collections of objects) or thin (removal of individual objects from a collection) the event content. D2/3PD are secondary (more refined and contain analysis data) and tertiary (i.e., NTUP format, organized in flat ROOT n-tuples files and suitable for plotting final results) Derived Physics Data.
- TAG data are event-level metadata (thumbnail information about events to support efficient identification and selection of events of interest to a given analysis) and stored in a relational database.

1.2.3.3 The Operational Model

The ATLAS experiment has a hierarchical model for data production and distribution. The primary event processing occurs at CERN in the Tier-0 facility (CERN Analysis Facility, CAF). The RAW data is archived at CERN and copied (along with the primary processed data) to the Tier-1 facilities around the world. These facilities archive the RAW data, provide the reprocessing capacity, provide access to the various processed versions and allow scheduled analysis of the processed data by physics analysis groups. Derived datasets produced by the physics groups are copied to the Tier-2 facilities for further analysis. The Tier-2 facilities also provide the simulation capacity for the experiment, with the simulated data housed at Tier-1s. In addition, Tier-2 centers provide analysis facilities and some provide the capacity to produce calibrations based on processing some raw data. The CERN Analysis Facility pro-

vides an additional analysis capacity, with an important role in the data-intensive calibration and algorithmic development work [3].

1.2.3.4 Modern Architecture of the Operational Model

Historically the ATLAS experiment used a strictly hierarchical cloud model. At the time when the original computing model was defined, sufficient network connectivity was only guaranteed within the national research networks and on the dedicated links between the Tier-1 computing centers. Therefore, static regional groupings (known as *clouds*) were defined, each consisting of one Tier-1 and several Tier-2s. The computing model also constrained jobs to be executed in those computer centers where the data was locally available.

The new model is about to relax the boundaries of the old computing model a step further, adding dynamic cloud configurations defined as follows [5]:

- *Nuclei* are made up of Tier-1s and selected Tier-2s, which pass certain qualifying criteria. These sites will be the center of a temporary cloud. Tasks will be assigned to them and task outputs will be aggregated in the nuclei.
- *Satellites* will be made up of other Tier-2s in the cloud that provide the processing capacity. Satellites can be assigned to nuclei outside their regional cloud.

In the setting of the ATLAS data management, each Tier-1 hosts a WLCG FTS (File Transfer Service) to handle file transfers within and into its representative association, while the transfers to and from the Tier-0 are managed by another FTS instance directly from CERN. Tiers association also forms the structure around which the management of the ATLAS computing activities is organized.

1.3 Related Work

This dissertation focuses on the analysis of user activities and interests to discover data, and influence user behavior aiding data distribution in the high-performance Workload Management System (WMS) PanDA (Production ANd Distributed Analysis system). In order to help the reader understand some of the later detail, in this section we will describe the current organization of the data management and distribution, and of the management of workload for data processing.

1.3.1 The ATLAS Distributed Data Management System

The management of the massive amount of data (hundreds petabytes) distributed over the storage facilities of the computer centers in the WLCG relies on the Distributed Data Management system called Rucio [6] in the ATLAS experiment. Rucio is the latest generation of the DDM system (it is the successor of DQ2, Don Quixote 2) with a scalable and reliable data organization and placement property.

The distribution of data also relies on the policies defined by the Computing Resource Management (CREM) and is guided by the ATLAS Computing Model (within the operational constraints). Thus, such parameters as proposed data lifetime, data type, and the minimum number of data replicas are also considered in a decision making process on where to store the data (Tier-1 or Tier-2, disk or tape storage).

Features of Rucio include a unified dataset/file catalogue, a global namespace with a deterministic translation to local file names, the consequent removal of the need for local file catalogues [2], inter-operations with different Grid resources, enforcement of management policies like the ATLAS Computing Model, and enforcement of access controls, management of user and group quotas, and accounting of data replicas. Figure 1.4 [10] shows main DDM components and related services. The following

[10] http://rucio.cern.ch/overview_Architecture.html

subsection will describe the most significant concepts and features of the DDM system Rucio.

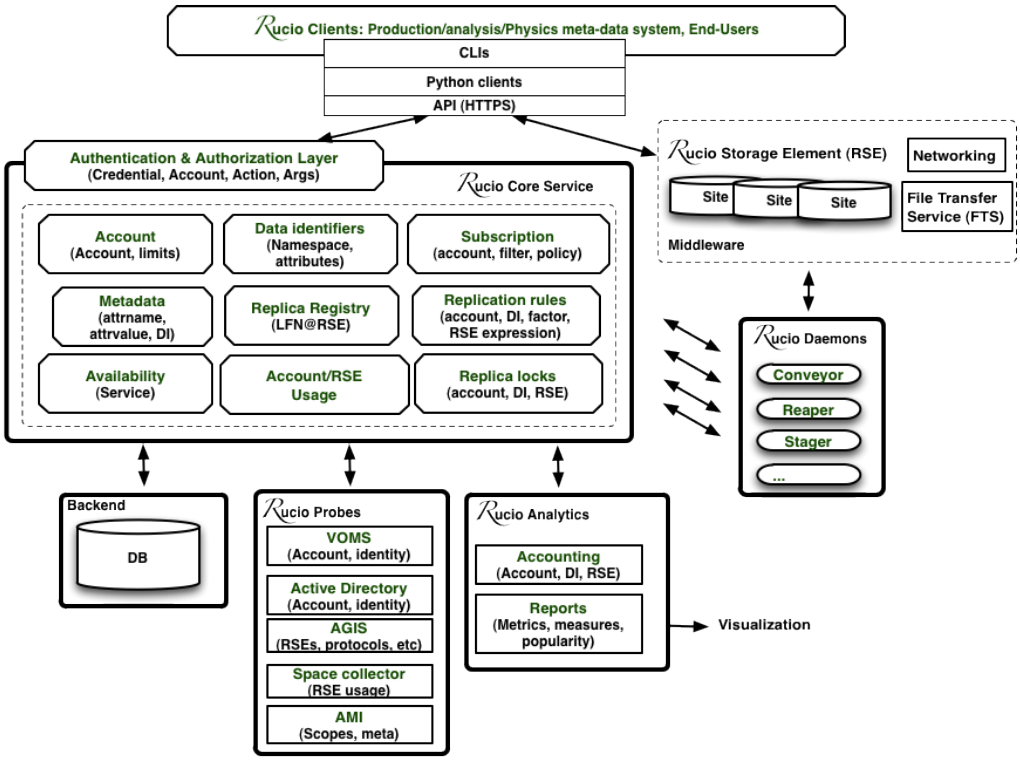


Figure 1.4: Overview of Rucio Architecture

1.3.1.1 Key Concepts

A DDM *account* represents every user, group, or organized activity at the ATLAS experiment, and associates them with corresponding permissions, (storage) quotas, and data namespace (which is called *scope*) [7].

Data management supports different types of data structures, that are referred as *data identifiers* (DID): files, datasets, and containers (aggregations of datasets or other containers) [6]. Thereby a *file* is the smallest operational unit of data.

Every DID is unique and defined as a tuple made up of a scope and a name. A DID name is unique within a scope, and can be used in different scopes at the same time [6].

Rucio Storage Element (RSE) is a logical concept to address and utilize storage systems in the Grid infrastructure. RSE is a container of physical files (file replicas) and represents a unit of storage space within Rucio. It is described by unique name, access protocols, host names, storage type, space availability, quality of service, etc. There are several protocols that are used by Rucio to interact with the storage systems, in particular WebDAV, or Amazon S3.

The physical paths of files (*Physical File Name*, PFN) are obtained via a deterministic function of the scope and name of the file (*Logical File Name*, LFN). The PFN is prefixed by the protocol and the site specific part (that are obtained from the internal Rucio catalog) to give a *Uniform Resource Identifier* (URI) [8]. Every file replica is associated with a corresponding URI.

1.3.1.2 Data Distribution

Rucio uses replication rules and a subscription mechanism for replicas management. A replication rule describes how a particular DID must be replicated on a list of RSEs. The minimum number of replicas (that satisfy the rule) will be determined to optimise the usage of the storage space, to minimize the number of data transfers, and to enable automated data distribution. A subscription is defined as a replication policy which is based on metadata for DIDs that will be produced in the future. Thus, Rucio will generate a rule for every DID that matches the parameters of the subscription, and based on this rule, it will create corresponding replicas [6].

1.3.1.3 Rucio Components

This subsection will provide a short overview of the main components of the current DDM system Rucio. The reader is referred to Figure 1.4 for the below descriptions.

The *Database* is a primary storage used to persist all the logical data; direct access is only granted to the Rucio server and its daemons (Rucio uses an ORM [11] approach).

The *Rucio Server* layer combines several Rucio core components together and offers a common, https-based, REST [12] API for external interaction (i.e., a passive component listening to incoming queries). The server is also responsible for authentication and authorization of clients, and for management of the request execution at the request-defined core component. Rucio core components are allowed to communicate with each other, as well as with the Rucio Storage Element abstraction.

Rucio Daemons represent agents that operate on user requests or requests made by the Rucio core asynchronously:

- the *Conveyor* is responsible for requests for data transfers,
- the *Reaper* is responsible for deletion of the expired data replicas,
- the *Undertaker* is responsible for obsoleting data identifiers with expired lifetime,
- the *Transmogrifier* is responsible to apply subscriptions on newly created or existing data to generate replication rules, and
- the *Judge* is the replication rule engine.

[11] ORM is an object-relational mapping that makes it possible to address, access and manipulate objects without having to consider how those objects relate to their data sources.

[12] REST (Representational State Transfer) is an architectural style for designing distributed systems. It is not a standard but a set of constraints, such as being stateless, having a client/server relationship, and a uniform interface.

The *Rucio Storage Element* (RSE) is an abstraction layer that is responsible for all interactions with different Grid middleware tools which interact with the Grid storage systems. It effectively hides the complexity of these tools and combines them into one interface used by Rucio. The abstraction layer is used by the clients, the server as well as the Rucio daemons.

The *Rucio Clients* layer offers a command line client for users as well as application programming interfaces which can be directly integrated into user programs. All Rucio interactions are transformed by the client into https requests which are sent to the REST interface of the Rucio server. Consequently, external programs can also choose to directly interact with the REST API of the server (e.g., by libcurl).

1.3.2 The Production and Distributed Analysis System

The Production and Distributed Analysis system (PanDA) is a workload management system that federates hundreds of heterogeneous computing centers of the WLCG into a unique job submission system and manages distributed resources intelligently [5] (the term *task* is applied for a set of jobs, that are combined based on same specifications, the term *job* is applied to a computational task or unit of work). PanDA is an automated yet flexible workload management system which can optimally make distributed resources accessible to all users. PanDA was originally developed for US physicists and adopted as the ATLAS wide WMS in 2008 (in use for all ATLAS computing applications).

Key features of PanDA are:

- pilot-based job execution system (i.e., there is a lightweight process scheduled on computing nodes that interacts with the core to schedule computing tasks);
- central job queue;

- fair-share or policy driven priorities for thousands of users and hundreds of resources;
- automated brokerage based on CPU and storage resources;
- automatic error handling and recovery;
- extensive monitoring;
- modular design.

The purpose of PanDA is to manage the data production process (manage central and group production of data for the overall ATLAS collaboration and/or a particular physics working group) and to manage analysis process.

1.3.2.1 PanDA Components

The objective of this section is to introduce the essential components of PanDA (Figure 1.5 [9]) with emphasis on their features that benefit the system and thus make PanDA flexible with on-demand access to distributed resources through a pilot mechanism.

The *PanDA Server* is a central hub that is responsible for jobs/tasks queue management [10], and communications with the pilot sub-system and DDM system. Its component systems are:

- Database backend is the primary storage for information about all submitted jobs, data and user descriptions.
- Task Buffer represents a job queue manager that keeps track of all active jobs in the system.
- Job Brokerage operates to prioritize and assign work on the basis of job type, priority, input data and its locality, software availability, and required resource capacity (e.g., CPU speed, memory, disk space, etc.).

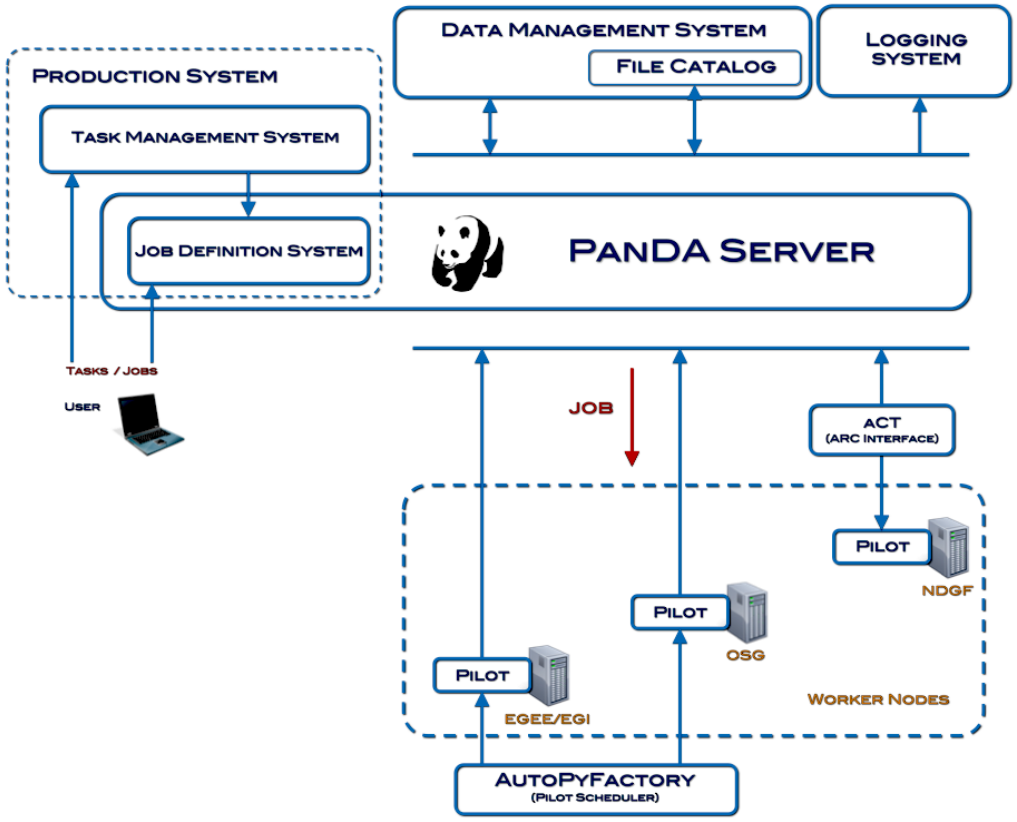


Figure 1.5: General Structure of the PanDA system

- Job Dispatcher dispatches job payloads upon pilot requests, and manages heartbeat and other status information coming from pilots.

The *Panda Pilot* represents the lightweight execution environment (effectively a wrapper) for PanDA jobs: to prepare the computing element, to request and receive the actual job payloads from the Job Dispatcher, perform setup, execute the payload, i.e., run the jobs themselves, and cleanup work surrounding the job [11]. In the context of PanDA Pilot:

- Job wrapper is a process that is responsible to copy the input files, setup the runtime environment, execute the job payload, transfer the job output files

(either by itself or by delegating this task to the DDM system) and finally to perform cleanups.

- Pilot Factory represents an independent and automatic system for the pilot dispatch and control. Particularly, PanDA uses *AutoPyFactory* that is a pilot submission, management and monitoring system. (Supersedes first generation PandaJobScheduler, as well as the second generation system, the AutoPilot.)

JEDI (Job Execution and Definition Interface) adds the capability to accept work defined as high level tasks, and breaks these down optimally based on the dynamic state of the the available resources [5]. Traditional workload management systems use jobs as the atomic unit and define these at the level of input files. JEDI on the other hand can handle bookkeeping at different granularities: task, job, file and event ranges.

PD2P (PanDA Dynamic Data Placement) is an intelligent subsystem of PanDA and is responsible for the dynamic management of data replicas (e.g., adding new replicas of popular datasets). PD2P makes secondary copies at corresponding sites by taking the following factors into account: popularity, locality, the usage pattern of the data, the distribution of CPU and storage resources, network topology between sites, site operation downtime and reliability.

Other general use components are, e.g., the *Information system*, and the *Monitoring system* (*PandaMonitor* is a web application to monitor jobs execution and to manage complex tasks, defining steps and dependencies between the jobs inside a task).

The latest system developments in PanDA move towards a finer granularity in the job definition, together with automated tuning of job settings. This is partially done through the addition of the JEDI component, which tracks work not only at task

and job level, but also at file and event range level, in order to generate tailored jobs for each compute resource. The *Event Service* is based on the JEDI capabilities and extends these by providing a continuous stream of short jobs to unpledged resources and takes care of staging out the output into a highly scalable Object Store to avoid any losses. The Event Service is designed to fully exploit opportunistic usage by decoupling the processing from the bulkyness of files and instead streaming inputs and outputs to/from the worker. The Event Service lowers the granularity of jobs to the event level and allows an almost continuous event streaming to the worker nodes that write their output quickly and hence have a negligible loss if the worker vanishes [5].

1.3.2.2 PanDA Workflow

The job submission on the grid is done in multiple steps that involve both the WMS - PanDA and the DDM system - Rucio. Job specifications (with corresponding input data) are defined by users (or are taken from the corresponding *task*) and are then submitted to PanDA.

The PanDA server receives jobs into the task queue (Task Buffer), upon which a brokerage module prioritize and assign work (cooperated with DDM system to determine the replicas location of the jobs input data, its availability, the current workload, etc.). The AutoPyFactory pre-schedules pilots to computing sites of the corresponding Grid infrastructure using Condor-G (a Condor system extension that allows for jobs to be submitted over the Grid through Globus-enabled gatekeepers that bridge between sites across administrative domains). Pilots retrieve jobs from the PanDA server in order to run the jobs as soon as CPU slots become available. Pilots are designed to use resources efficiently; they exit immediately if no job is available and the submission rate is regulated according to workload. Each pilot executes a job

on a worker node (WN), detects zombie processes, reports job status to the PanDA server, and recovers failed jobs. For NDGF, the ARC [13] control tower retrieves jobs from the PanDA server and sends the jobs together with pilot wrappers to NDGF sites using NorduGrid/ARC middleware [10].

An independent subsystem inside PanDA manages the delivery of pilot jobs to WNs, using a number of remote (or local) job submission methods. A pilot once launched on a WN contacts the PanDA Job Dispatcher and may receive a pending job appropriate for the site/node. An important attribute of this scheme for interactive analysis (where minimal latency from job submission to job launch is important) is that the pilot dispatch mechanism bypasses any latencies in the scheduling system for submitting and launching the pilot itself. The pilot job mechanism isolates workload jobs from grid and batch system failure modes (a workload job is assigned only once the pilot successfully launches on a WN). The pilot also isolates the PanDA system from grid heterogeneities (which are encapsulated in the pilot), so that at the PanDA level the compute locations appear homogeneous.

Workflow is different between production and analysis type jobs although PanDA is designed for both production and analysis to use the same software, monitoring system, and facilities.

For production, PanDA receives centrally defined tasks on the basis of physics needs and resource allocation in ATLAS. Tasks are assigned to clouds (logical groupings of computer centers) based on the amount of disk space available on the Tier-0/1 storage element, the locality of input data, available CPU resources, the Memorandum of Understanding (MoU) share which specifies the contributions expected from each participating institute, and downtime of the Tier-0/1. Tasks are automatically

[13] ARC (Advanced Resource Connector) provides a reliable implementation of the fundamental grid services, such as information services, resource discovery and monitoring, job submission and management, brokering and data management and resource management.

converted to many jobs for parallel execution. Once a task is assigned to a cloud, jobs are assigned to computing centers in the cloud. After jobs successfully finished on WNs, output files are aggregated back to the Tier-1 from Tier-2 [10].

ATLAS production jobs are roughly categorized into three groups of activities: MC simulation, data reprocessing, and physics working group production. Generally jobs for data reprocessing and physics working group production have higher priorities than jobs for MC simulation because of urgent needs for the former activities [10].

For analysis, each end-user submits a user task (job set) that is split to multiple job subsets according to localities of input datasets, workload distribution, and available CPU resources at sites. A job subset is sent to a site where input files are available, i.e., if input files are distributed over multiple sites there will be multiple job subsets and they will be sent to multiple sites. Each job subset is composed of many jobs. One of the most significant differences between production and analysis is policy for data transfers. The DDM system transfers files over the grid for production, while analysis does not trigger file-transfers since analysis jobs are sent only to sites where the input files are already available. This is mainly because analysis jobs are typically I/O intensive and run on many files. In order to demonstrate the magnitude of jobs executed, we summarized the total number different jobs (with emphasis on analysis jobs) submitted in Table 1.1.

Table 1.1: Numbers of processed jobs at PanDA (for the last five years)

Year	Number of all jobs	Number of analysis jobs	Number of success analysis jobs
2011	222,212,837	122,133,639	91,565,969
2012	297,627,703	176,103,863	122,078,104
2013	340,827,067	207,355,412	142,496,850
2014	316,563,177	185,787,693	135,780,198
2015	371,808,032	206,388,996	140,519,048

1.4 Organization of the Dissertation

The rest of this dissertation is organized as follows. Chapter 2 will review previous work that propose approaches to improve the efficiency in DDM and WM systems (approaches for corresponding ATLAS systems are considered as well). These approaches are presented according to the methods that they use (e.g., machine learning algorithms to predict the next state of data popularity). Chapter 3 provides the background on techniques and methods that were used in the analysis of PanDA data and in the synthesis of the personalization solution. Chapter 4 will present the initial study that was conducted to explore data from PanDA system to find usage patterns and to utilize it in a simulation prototype of a recommender system. This study motivates the need of a recommender system for PanDA. Chapter 5 will present a system called Data Watcher, based on recommender system approach, that can find correlations among users, and users and data use. Chapter 6 will provide an analysis of the capabilities of the Data Watcher system and its preliminary results. Finally, Chapter 7 concludes the dissertation and outlines future directions for extensions and research.

CHAPTER 2

LITERATURE REVIEW

This section describes the approaches that have been proposed in data- and workload-management systems, to improve the operational processes in management of the vast amounts of data and computing resources. We will focus our investigations on approaches that include machine learning techniques, as these techniques are currently seen as the most promising extensions. The following literature review presents the most closely related work in terms of grid infrastructure based distributed computing systems.

Before we delve into our literature review, we would like to mention why our research direction moved towards data mining/machine learning techniques. In one of our previous studies [12] (that is related to the problem of the current research but is not a part of it) we estimated the popularity of data in a WMS using stochastic methods. The primary intention was to build a Bayesian network that would help to calculate corresponding probabilities for data popularity. Data that was used for the analysis is input data for PanDA jobs (i.e., datasets that are managed by the ATLAS DDM system), and only those analysis jobs that finished successfully were considered. The analysis required a highly complex model design; yet it only showed negligible correlation between datasets and actual user interests. The methodology did not consider relationships between different users' past datasets but was based on pure popularity analysis. Based on such results our direction of research was changed towards the study of user personalization.

2.1 Data Management

Every computing system has specific components that are responsible for data management, i.e., data organization and handling. Here approaches to improve the efficiency of data distribution based on machine learning will be reviewed. The goal of such improvements is to predict the popularity of data and to use the outcome to replicate data to the computing nodes where it will be most desirable. All reviewed approaches are focused on calculating the probabilities for a defined data being kept on a particular storage based on the past data usages, but to the best of our knowledge ours is the first work using user models in a WMS that could describe preferences of each user (and thus improve estimated weights).

2.1.1 Data Popularity Prediction for ATLAS DDM

We introduced the ATLAS DDM system Rucio in an earlier section (Section 1.3.1). Rucio is responsible for the organization and management of the data from detector, simulation data, and all of the derived physics data (which is in use by the ATLAS collaboration).

[13] and its companion work [14] propose a dynamic way of pro-active management of data replicas (remove replicas of data with zero user interest and add extra replicas of data with increasing user interest). This approach is based on predictions of future data (i.e., datasets in ATLAS terms) accesses by analysing the popularity of data in the past (such information includes number of accesses, files, involved users and computing nodes) with focus on datasets that are used in user analysis jobs in PanDA (production jobs are disregarded). Thus Beermann et al. attempt to forecast the level of the popularity of data.

The goal of the proposed tool is to estimate the number of dataset accesses during a certain time period in the near-term future. The implementation relies

on the training and use of an artificial neural network (ANN) to find trends in the historic dataset accesses (i.e., common dataset access patterns). Thus, the ANN uses aggregated accesses for each dataset on the Grid per week as an input data (i.e., n input neurons for n weeks, $A_{1,2..n}$), and produces an estimated number of the accesses for the next week as the output (i.e., the output neuron provides the predicted number of accesses for the $(n+1)^{th}$ week, A_{n+1}). This approach also takes into consideration that datasets with different attribute sets have different access patterns (attributes that are considered for data partitioning are dataset project and data format), thus each set of attributes has its own ANN to process datasets.

Evaluation of this approach is done with comparison to the static and linear prediction methods, and shows a significant advantage of neural networks at picking up the trend of dataset accesses in the future, i.e., predicting if the popularity of a certain dataset goes up, down or stays the same from one week to the next.

The efficiency of the data distribution with the usage of the predicted popularity was evaluated with the help of a simplified grid simulator. Preliminary results of the simulation of workload based on two sample weeks (to perform data processing at the set of computing centers that represents one cloud, in terms of the ATLAS Computing Model) showed that extra replica(s) of datasets predicted to be popular reduced the job waiting time for the data processing and decreased the number of jobs waiting in the queues for job slots. Thus, the simulation efforts demonstrated that the data redistribution following the prediction leads to a better usage of the available resources.

2.1.2 CMS Popularity Prediction for Dynamic Data Placement

The CMS experiment is another experiment at the LHC (see Section 1.2). Despite using the same WLCG infrastructure, CMS' management systems and workflow

organization is different from what the ATLAS experiment provides. Here we describe a new approach that was proposed by [15, 16] for the CMS data management system (as a part of a project that aims to build adaptive data-driven models of CMS Data Management and Workload Management activities).

CMS data management is organized into the following systems: PhEDEx (Physics Experiment Data Export - a data placement system), DBS (Dataset Bookkeeping System - provides the means to define, discover and use CMS event data), Dashboard (global monitoring system which keeps tracks of user jobs), SiteDB (database for site pledges, deployed resources, and manpower onsite), PopDB (CMS datasets popularity database), DDM (Dynamic Data Management system) and others.

The goal of this approach is to model and predict CMS dataset popularity by using machine learning (ML) algorithms based on historical usage of datasets via various available meta-data information, and to use these predictions for the Dynamic Data Management system (thus providing the foundation of data-driven approach for the CMS computing infrastructure). As the result, the DCAFPilot (Data and Computing Analysis Framework Pilot) framework was developed. This framework collects (and transform) structured and semi-structured information about data description and data usage extracted from CMS data-services (with the focus on particular type of user based data). It uses ML classifiers such as i) RandomForest, LinearSVC, SGDClassifier from *scikit-learn* (machine learning library for the Python programming language), ii) *Vowpal Wabbit* (online machine learning system library and program developed originally at *Yahoo! Research*, and currently maintained by *Microsoft Research*), and iii) *xgboost* (eXtreme Gradient Boosting package in R programming language, that includes efficient linear model solver and tree learning algorithm) to make a prediction of the data popularity (i.e., to predict decline in

popularity of certain datasets). Predicted values are used to allocate replicas of popular datasets at various computing nodes by the Dynamic Data Management system (to improve resource allocation and to reduce redundant activity), that is later compared with the actual data access patterns.

Preliminary results of this framework showed its high efficiency with the prediction of the data popularity and its utilization in data placement process, where the number of data replicas kept on Grid nodes is related to their popularity [17] (e.g., for a particular data type the true positive rate (i.e., sensitivity) was 0.97 ± 0.05, and true negative rate (i.e., specificity) was 0.99 ± 0.02).

2.1.3 Data Placement Optimization for the LHCb experiment

Optimization of the data storage utilization is one of the crucial tasks for the LHCb (see Section 1.2) experiment as well. [18] proposes an approach to use machine learning algorithms to predict future data popularity and its usage intensity; it helps in selecting datasets that should remain on disk storage (fast access, but the space is limited), and to reorganize data distribution (e.g., decrease or increase the number of data replicas). The data usage history and dataset metadata (such as the origin, detector configuration, files type, dataset type, event type, creation week, first usage week, last usage week, size for one replica, etc.) are used as an input to the machine learning algorithms. The provided data usage history in LHCb is relatively sparse, thus such approaches as artificial neural networks are not applicable as they would result in over-fitting.

The above approach uses three modules in order to pick the data that will be popular in future, to predict the level of its popularity (i.e., data usage intensity), and to adjust its number of replicas at the chosen storage locations (data replicas

regulation for the most popular datasets would reduce the average access time for that data):

- The Data Popularity Estimator module uses a classifier (a supervised machine learning algorithm *Gradient Boosting Classifier*) to predict data future popularity. The data popularity itself represents the probability for the dataset to be useful in future, thus unpopular datasets are removed from disk storage.
- The Data Intensity Predictor module uses non-parametric models to predict the dataset future usage intensities. Time series analysis and regression algorithms (*Nadaraya-Watson kernel smoothing* and *rolling mean values*) are used to make these predictions. Input information for this module is the dataset usage history.
- The Data Placement Optimizer module uses a loss-function that considers the cost of storage of the data on disk, the cost of storage of the data on tape, the cost of mistakes (when a dataset was removed from disk, but then is used). As the result the loss function (its optimization process) helps to find the data popularity threshold value and the optimal number of replicas for a dataset with predicted usage intensity (e.g., increase of the number of replicas for the popular data will reduce their average access time).

As the outcome of the above method, the distribution of data on disk storage was reorganized such that it released unpopular data and increased the number of popular data, thereby reducing the average access time. As comparison, the Last Recently Used (LRU) algorithm showed slightly reduced performance in space savings, but had a significantly higher number of wrongful removals (number of datasets which are proposed to be removed from disk storage, but are then used again in future).

2.2 Workload Management

2.2.1 General Concepts

The Workload Management System (WMS) is responsible for distributing and managing computational tasks across distributed computing infrastructure (represented by computing elements (CE) and storage elements (SE)) in order to provide optimal processing performance for defined requirements and to maintain the load balance across the entire infrastructure. The management of a workload is deemed "intelligent" when it can self-identify necessary processing needs and security protocols; determine the capacity at which it can best perform its given tasks; and is fully integrated into business services in such a way that its processing work does not interfere with other organizational computing efforts.

Matteo et al. [19] provide a comprehensive analysis of Pilot-Job based systems, which are the quintessence of WMSs. There are five representative features of Pilot-Job systems:

- *Task-level distribution* and *parallelism* on multiple resources;
- *Master worker/node* to coordinate the processing and execution of the computational tasks (i.e., jobs);
- *Multi-tenancy*, that defines the way resources are used (e.g., queues organization for efficient and fair resource sharing, like *Job Schedulers*);
- *Multi-level* scheduling strategy to increase the granularity level of control (e.g., a global scheduling decision results from a set of local scheduling decisions);
- *Resource placeholders* that decouple the acquisition of compute resources from their use to execute the tasks of an application.

In general, Pilot-Job abstraction generalizes the reoccurring concept of utilizing a placeholder job (which instance is commonly referred to as Pilot-Job or pilot) as a

container for a set of compute tasks [20]. Pilot-Job systems are characterized by the following concepts:

- using late binding to utilize resources dynamically (i.e., the workload is distributed onto resources only when they are effectively available); and
- decoupling the workload specification from the management of its execution (i.e., scheduling of workloads on selected resources).

2.2.2 Pilot-Data Abstraction

Pilot-Data (PD) represents an extension of the Pilot-Job abstraction for supporting the management of data in conjunction with compute tasks (for data-intensive applications) [21]. Thus PD separates logical compute and data from physical resource providing efficient placements of compute/data in a way it would not be dependent on the underlying infrastructure. The following capabilities of PD are emphasised by André et al. [21] as key characteristics:

- Dynamic resource management with a unified access layer to different heterogeneous data computing infrastructures;
- Distributed namespace for data;
- Higher-level abstraction for compute/data coupling;
- Compute/data scheduling (i.e., data-aware workload management service).

The concept of Pilot-Data was implemented as an extension in BigJob [22]. BigJob is a Pilot-Job system which implementation is based on SAGA (SAGA is a programming system that provides a high-level API for accessing distributed resources. It provides the building blocks to develop the abstractions and functionalities to support the characteristics required by distributed applications whether directly, or as tools in interoperable and extensible fashion). BigJob natively supports parallel

applications (e.g., based on MPI, Message Passing Interface) and works independent of the underlying Grid infrastructure across different heterogeneous backend [22].

The PD abstraction raises an essential question that every WMS should be able to answer: whether to assign and move computational tasks to where data is located, or to move data to where computational tasks can be executed. Thus, "intelligence" in WMSs relates to the efficient answer of the above PD question. However, WMSs could benefit from additional intelligence (in the form of machine learning techniques) for example by being able to monitor user behavior in order to make scheduling decision; this is what our work is focused on.

2.3 Summary

The approaches reviewed in this chapter significantly extended the capabilities of data- and workload-management systems of scientific computing platforms. Proposed solutions for the data popularity prediction showed a demand for parameters that can describe future data behavior, and that can be represented as trustful metrics for data distribution. The diversity of the investigated methods (ANN, ML classifiers, regression analysis, etc.) allows for the adjustment of the input parameters in order to enhance the quality of the predictions.

WMSs rely on rational data distribution, and provide appropriate metrics (such as waiting time, processing time, brokerage weights, etc.) to the data management system that help in reducing the operational burden of the WMS workflow. Proposed abstractions for Pilot-Job and Pilot-Data approaches describe the general architecture of a WMS with an emphasis on intelligent data- and computing-resource utilization.

The lack of user-centric approaches (that could be able to resolve the problem of finding reasons for data popularity, and to utilize the obtained information in building

relational models) opens possibilities for further improvements in data management for storage and processing resource utilization.

CHAPTER 3

METHODOLOGY

Research objectives revolve around the discovery of (hidden) relationships between objects in distributed computing systems; thus, methods and techniques were taken from data mining field with an emphasis on machine learning. The goal of our research is to build user model that would reflect user interests and preferences in the usage of data. It includes the investigation of such processes as data extraction and filtering, model creation based on selected and transformed data, management of models (comparison, grouping, etc.), and prediction (with estimation of the corresponding weights) of model behavior.

The user model is then built based on personalization techniques as data mining processes. An essential component of the proposed approach is a recommender system (information filtering) with its implementation as collaborative and content-based filtering. Methods used in recommender system approaches represent predictive analytics that can provide estimated ratings for users in particular data and thus can reveal corresponding user interests and their significance in real time. The rating subsystem assists in collecting explicit user feedback to adjust the quality of the estimated data popularity (i.e., comprehensive data rating) and data significance metrics (i.e., objects' weights).

The methods and techniques applied in this research embrace different aspects of reaching the goals by considering not just conditional parameters of the context, but also by focusing on the content of objects and their projections (compared to others). Thus the content of data (i.e., the data description) is represented by a set

of attributes (i.e., data features) that are used to form user profiles to describe user preferences (and to justify the interest in a particular data). Such information would also be considered in grouping data based on representative attributes (i.e., attributes that characterize data at the highest extent). Another class of the applied methods is focused on object behavior within objects; e.g., the activity of a particular user (i.e., timeline of the data usage) is compared to activities of other users in order to discover usage patterns. Such methods do not need the content of data but rather the history of data access by a group of users.

The quality of the personalization process depends on comprehensive data analysis that reveals the reasons of user interest in a particular data and defines the significance of that data to the user (compared to other users and data).

The rest of this chapter will provide the background information needed i) to understand various technologies used to analyse PanDA data, and ii) to be able design our system called Data Watcher. More precisely, the following related techniques will be reviewed: data mining and recommender system basics, as encompassed by collaborative and content-based filtering approaches.

3.1 Data Mining

Data mining, in general, is considered as the analysis step of the *Knowledge Discovery in Databases* (KDD) process, and is defined as the application of specific algorithms for extracting patterns from data. Figure 3.1 provides a high-level description of the main steps in the KDD process [23].

Data mining functionality is used to specify the kind of patterns to be found in data mining tasks. In general, data mining tasks can be classified into two categories: descriptive and predictive. *Descriptive* mining tasks are aimed at characterizing the general properties of the data in the database. *Predictive* mining tasks on the other

hand perform inference on the current data in order to make predictions [24]. Our focus is on both descriptive and predictive mining tasks. Thus the goal of our mining process is to capture, model, and analyze the behavioral patterns and profiles of users interacting with data.

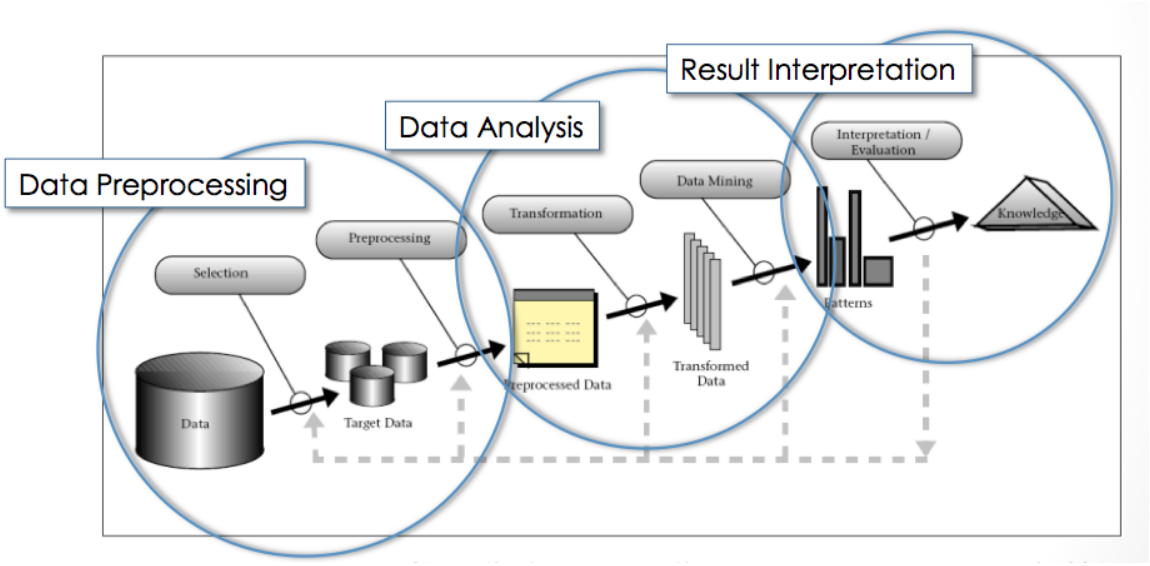


Figure 3.1: High-level overview of the KDD process

As Figure 3.1 shows, the three main cumulative steps of KDD are *Data Preprocessing*, *Data Analysis*, and *Result Interpretation*. The background on data analysis techniques that are employed is presented in the remainder of this chapter while the two other steps will be presented in Chapter 4.

In order to shed more lights on the purpose of data analysis techniques, we provide the following highlights in the context of our own research. The initial analysis described in Chapter 4 is focused on the discovery of associations and correlations within PanDA data. More precisely, the goal of that analysis is to detect frequently appearing patterns, i.e.,

- *frequent itemsets* (sets of items that frequently appear together in a transactional data set);
- *frequent sequential patterns* (e.g., frequently occurring subsequences, ordered subsequences of itemsets that appear in certain sequences of transactions);
- *generalized frequent structured patterns* (where data substructure can refer to different structural forms, such as graphs, trees, or lattices, which may be combined with itemsets or subsequences).

3.1.1 Association Analysis

The task of association analysis is to discover association rules by mining frequent itemsets. In general, the goal of association analysis is to discover hidden relationships in large data sets; we will denote the complete data set by $\mathbb{I}$ (set of items). Thus our objective is to identify item pairs that co-occur more frequently than a value predetermined by the analyst.

A rule is defined as an implication of the form $X \Rightarrow Y$ (a.k.a., the *if-then* rule), where $X, Y \subseteq \mathbb{I}$ and $X \cap Y = \varnothing$. Association rules are rules, which surpass analyst-specified *minimum support* and *minimum confidence* thresholds. The support $supp(X)$ over the set of items X determines how often the rule is applicable to a given data set, and is defined as the fraction of transactions (groups of elements from $\mathbb{I}$) that contain the defined set of items: $supp(X) = \frac{|\{t_m \mid X \subseteq t_m,\ t_m \in \mathbb{T}\}|}{|\mathbb{T}|}$; where $\mathbb{T}$ is a set of subsets from $\mathbb{I}$. The confidence of a rule $X \Rightarrow Y$ determines how frequently items in Y appear in transactions that contain X, i.e., $conf(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)}$. Therefore $X \Rightarrow Y$ will be used as an association rule if it satisfies analyst-predefined minimum support (σ) and minimum confidence (δ) limits:

$$\begin{aligned} supp(X \cup Y) &\geq \sigma \\ conf(X \Rightarrow Y) &\geq \delta \end{aligned} \tag{3.1}$$

Another measure for quantifying association rules is *lift*:

$$lift(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)supp(Y)} \tag{3.2}$$

and can be interpreted as the deviation of the support of the whole rule from the support expected under independence of both sides of the rule. Greater lift values ($\gg 1$) indicate stronger associations.

3.1.1.1 Association Rule Mining Algorithms

A common strategy for association rule mining algorithms is to decompose the problem into two major sub-tasks [25]:

- *Frequent Itemset Generation*, where the objective is to find all the sets of items that satisfy the minimum support threshold (these sets are then called *frequent itemsets*);
- *Rule Generation*, where the objective is to extract all the high-confidence rules from the frequent itemset generation step (these rules are then referred to as *strong rules*).

Frequent Itemset Generation is based on the transitive property of itemsets that every sub-itemset of a frequent itemset is also frequent, i.e., if $X, Y \subseteq \mathbb{I}$ and $X \subseteq Y$ and Y is a frequent itemset then X is frequent itemset as well; this implies that $supp(X) \geq supp(Y)$ (*anti-monotone* property of support). This property helps to eliminate the need to evaluate supersets that contain non-frequent itemsets. A frequent itemset is called a *maximal itemset* if it does not have a frequent superset.

Most associative rule mining algorithms are based on the *Apriori* [26] algorithm. Apriori is an iterative algorithm based on the above described properties and uses so-called *prior knowledge* discovered at each iteration. It operates by constructing *candidate* k-itemsets, filtering these candidates based on frequency, and using the results to explore and construct $(k+1)$-itemsets (supersets over the previous candidate k-itemsets). These steps repeat until there are no more new candidates.

3.1.2 Sequential Pattern Mining

The sequential pattern mining technique encompasses the mining of frequently occurring patterns ordered by time (i.e., ordered events). In our case these are the so called *paths* that users follow (data-sequences). The problem is to find all or the longest (with the most number of items and/or with the most number of transactions, i.e., internal groupings of items) sequential patterns within an analyst-specified *minimum support*, where the support of a sequential pattern relates to the number of data-sequences that contain the pattern (patterns are subsequences).

Association rules indicate intra-transaction relationships, while sequential patterns represent the correlation between transactions. They help to reduce the potentially large number of sequences into only the most interesting sequential patterns. To meet different user requirements, it is important to use a minimum support which prunes sequential patterns of no interest.

Sequential pattern mining algorithms are generally categorized into: Apriori-like algorithms, BFS (Breadth First Search)-based algorithms (e.g., GSP, MFS), DFS (Depth First Search)-based algorithms (e.g., SPADE, FreeSpan, PrefixSpan, SPAM), closed sequential pattern based algorithms (e.g., CloSpan, BIDE), and incremental-based algorithms (e.g., SuffixTree, FASTUP, ISM, ISE, GSP+, MFS+, IncSP, etc.).

For a more thorough taxonomy on sequential pattern mining algorithms, the reader is referred to [27].

In our case, performance (i.e., how long it takes to identify patterns) was not of utmost importance as our goal was to determine if there are significant patterns in our data. SPADE (Sequential PAttern Discovery using Equivalence classes) [28] was identified to best suit our needs as it lends itself for easy modifications (to satisfy our custom criteria). SPADE shows linear scalability with respect to the number of sequences. SPADE is based on lattice search techniques and provides the possibility to impose constraints on the mined sequences. The key features of SPADE include the layout of the database in a vertical id-list database format (the rows of the database consist of object-timestamped pairs associated with an event) with the search space decomposed into sub-lattices which are processed independently in main memory thus enabling the database to be scanned a maximum of only three times (sometimes just once on some pre-processed data). There are two search strategies used to find sequences in the lattices:

- BFS: the lattice of equivalence classes is explored in a bottom-up manner and all child classes at each level are processed before moving to the next.
- DFS: all equivalence classes for each path are processed before moving to the next path.

Using the vertical id-list database format, all frequent 1-sequences (initial sequences with only one element) can be computed over a single database scan. Computing the frequent 2-sequences can be achieved in one of the following ways; by pre-processing and collecting all 2-sequences above a specified lower bound, or by performing a vertical to horizontal transformation dynamically. Once this has been completed, the process continues by decomposing the 2-sequences into prefix-based

parent equivalence classes followed by the enumeration of all other frequent sequences via either BFS or DFS within each equivalence class. The enumeration process is the *union* or *join* of a set of sequences or items whose counts are then calculated by performing an intersection of the id-lists of the elements that comprise the newly formed sequence. By proceeding in this manner it is only necessary to use the first two subsequences lexicographically at the last level to compute the support of a sequence at a given level [28].

3.2 Data Mining for Personalization

Personalization is the process of making a system tailored to the needs and preferences of individuals. As the result of this process a system creates and maintains *user profiles*. User profiles are represented by a set of attributes categorized into: factual (that describe the user itself), and transactional/behavioral (that describe user actions). There are two different ways of creating user profiles that depend on who controls the creation: i) *manual*, i.e., *customization*, when user preferences are set by the user or operator, and ii) *automatic*, where information about user preferences is extracted automatically by the system (with minimal explicit control by the user). In this work the term personalization will be used more restrictively, i.e., only meaning automatic personalization.

User profiles should follow certain criteria and should be evaluated in terms of three different metrics:

- *popularity* - evaluates the significance of the attributes in the profile;
- *dissimilarity* - evaluates the distinctness of attributes in the defined profile;
- *parsimony* - evaluates the succinctness of the profiles described by a set of attributes.

Data mining is then the automated analysis of massive data sets, knowledge discovery from data (extraction of interesting, non-trivial, implicit, previously unknown and potentially useful patterns). Data mining techniques can be employed to discover a set of rules describing users' behavior, likewise these types of techniques can be applied for user profile management.

In our context, the goal of applied data mining techniques is to capture, model, and analyze the behavioral patterns and profiles of user interactions. The overall process of personalization based on data mining consists of three phases: i) data preparation and transformation, ii) pattern discovery, and iii) recommendation.

For data mining to function, we need to be able to determine how well objects relate to each other, i.e., how similar they are or what the distance between them is. This is paramount in order to be able to group objects. In the next sub-section we will recap some of the similarity and distance metrics that are usually employed.

3.2.1 Similarity and Distances

Similarity is the measure that shows how "close" to each other two objects are; the "closer" objects are to each other the larger the value of the similarity should be. *Dissimilarity* is the opposite to the similarity, it shows to what extent two objects are different (i.e., how "far" from each other two objects are). Distance metrics are dissimilarity measures that are never negative, and zero only if two objects are the same (i.e., $d(x, y) = 0$, if $x = y$).

1) *Euclidean distance*; this distance is by far the most well used distance measure in engineering and science (also sometimes referred to as the L2-norm).

$$d(x, y) = |x - y| = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2} \tag{3.3}$$

2) *Pearson's correlation coefficient* is a statistical measure of the strength of a linear relationship between paired data (covariance/correlation coefficients can only capture linear dependency between the variables);

$$pcc(x,y) = \frac{cov(x,y)}{\sigma_x \sigma_y} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{3.4}$$

3) The *cosine similarity* between two vectors (or two documents in a corresponding vector space) is a measure that calculates the cosine of the angle between that vectors;

$$cs(X,Y) = cos(\theta) = \frac{X \cdot Y}{\|X\|\|Y\|} = \frac{\sum_{i=1}^{n} x_i \cdot y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \tag{3.5}$$

4) The *Jaccard index*, also known as the *Jaccard similarity coefficient*, is a statistic used for comparing the similarity and diversity of sample sets.

$$js(X,Y) = \frac{|X \cap Y|}{|X \cup Y|} \tag{3.6}$$

The Jaccard coefficient measures similarity between sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets. Thus, this index only uses presence-absence data.

3.2.2 Classification

The process of mapping objects to a certain class or set of classes is called *classification*. *Supervised classification* maps objects to a set of classes that are known in advance. Contrary, *unsupervised classification* maps objects to classes that are

formed during the classification process, these classes are not known in advance; the term *clustering* is usually used to refer to such process. Most popular classifiers are Nearest Neighbors, Decision Trees, Rule-based, Bayesian, Artificial Neural Networks.

3.2.3 Clustering

Clustering is the process of finding structure in a set of *unlabeled* data, and thus it is considered *unsupervised learning*. Thus, a cluster is a collection/group of objects which are "similar", but are "dissimilar" to the objects belonging to other clusters. The similarity measure used depends on the type of data. If similarity criterion is distance (*distance-based clustering*), then two or more objects belong to the same cluster if they are "close" according to a given distance (e.g., euclidean distance). If objects are grouped according to their fit to descriptive concepts (not according to simple similarity measures) then we refer to the process as *conceptual clustering*. In this work we consider only distance-based clustering.

Clustering algorithms can be categorized based on how the underlying models operate; they can be divided into three categories:

- Partitioning methods (or centroid clustering), create a predefined number of partitions (clusters) over a given data set:
  - Exclusive clustering: data are grouped in an exclusive way, so that if a certain entity already belongs to a cluster then it can not be included in another cluster (e.g., k-means clustering, Lloyd's algorithm);
  - Overlapping clustering: uses fuzzy sets to cluster data, so that each point may belong to two or more clusters with different degrees of membership, thus data will be associated with an appropriate membership value (e.g., Fuzzy C-means clustering);

- Hierarchical methods create a hierarchy of clusters; they can be:
  - Divisive, i.e., top-down approach; starts from the whole data set of items as a single cluster and recursively partition this data;
  - Agglomerative, i.e., bottom-up approach; starts from individual items as clusters and iteratively aggregates smaller clusters;
- Model-based methods that use a mathematical model to fit data points into clusters (e.g., a probability distribution, thus clusters are defined based on how likely the objects included belong to the same distribution).

A good cluster model creates high-quality clusters with high intra-cluster similarity and low inter-cluster similarity. The quality of the results however depends heavily on i) the nature of the data; ii) the similarity measure used by the method; iii) the actual implementation of the method; and iv) user supplied parameters (e.g., the number of clusters selected).

3.3 Recommender Systems Background

A recommender system uses a set of machine learning/data mining processes, that aim to guide users in a personalized way to interesting or useful items in a large space of possible options [29]. Frequently cited examples are Amazon's personal product recommendations or Netflix's personal movie recommendations, each of which is based on the history of the users behavior, the behavior of "similar" users, and product similarities. *Users* and *items* are two basic classes of objects that recommender systems deal with. Each object is represented as a set of attributes (i.e., features), where an attribute is defined as a property or characteristic of an object [30]. Recommender systems predict the user's rating (likeliness score) for certain items, and/or provide a set of items that could be interesting to the user. Prediction thus can

be expressed as a numerical value, representing the likeliness of the particular user's interest in an item that the user has never chosen before. Similarly, recommendation can then be defined as a list of (certain number of) items that have the highest predictions of user interest (i.e., top-N recommendations). In general, recommender systems help their users to decide on appropriate items and ease the task of selecting the most suiting items from a possibly humongous collection.

In general, there are two substantially different sets of algorithms employed in recommender systems: i) collaborative filtering algorithms, and ii) content-based filtering algorithms. A recommender system can employ algorithms from either or both of these sets. The choice of the actual technique deployed will heavily depend on the domain where the system will be used. Notwithstanding the type of the algorithm used, recommender systems rely on data collected from all users. Internally, recommender systems keep their data in a set of user-item pairs that form an utility matrix, "giving for each user-item pair, a value that represents what is known about the degree of preference of that user for that item" [31].

The accuracy of a recommender system depends on relevant feedback forming statistically significant user preferences; such feedback can be either explicit or implicit. Explicit feedback provides a more clear understanding of user preferences, as users will be explicitly asked to rate items; thus such feedback can be represented by ratings and reviews. Implicit feedback, on the other hand, is feedback that is inferred from users' interactions and activities and is computed into a rating by an algorithm. Without loosing generality, we will restrict the representation of feedbacks to "ratings" (i.e., no textual reviews as feedback so as not to have to focus on natural language processing). Ratings thus can be classified into following classes: numerical or scalar, binary, and unary. A numerical rating is represented by a scalar from a discrete or a continuous rating scale with a limited range (e.g., discrete number of

"stars" for products on Amazon). A binary rating is expressed on two states of user preferences: like or dislike. An unary rating is one possible rating value that mostly shows only positive relation between user and item (i.e., it does not capture user's indifference or dislike).

3.3.1 Collaborative Filtering

Collaborative filtering (CF) provides recommendations and predictions based on behavior and/or ratings of other users in the system. As the basic approach, it calculates similarity metrics between users; and thus it works on the premise that similar users will behave similarly (i.e., most likely will give similar ratings for the same items). In other words, CF algorithms provide recommendations based on users previous preferences and the opinions of other, like-minded users.

CF comes in two major flavors: memory-based and model-based. Memory-based approaches keep all data to be processed in the memory and perform calculations at the time of generating recommendations. Model-based approaches perform the computationally expensive learning phase offline (i.e., learn a predictive model) [29]. Memory-based approaches can then be further subdivided into user-based (relies on groups of similar users) and item-based (similar items) categories, both of which use the entire utility matrix to make predictions/recommendations. In the user-based approach, the system forms user groups based on a similarity measure between their ratings and provides recommendations based on the items liked by the k-nearest neighbors (the k most like-minded users). In the item-based approach, for each item, the system generates a set of similar items (items with the highest similarity measure, i.e., top-k similar items) and then considers other users ratings to these items to provide the corresponding recommendations. The most widely used

similarity measures are Pearson correlation coefficient, cosine similarity, and Jaccard index (a.k.a., Jaccard similarity coefficient).

The model-based approach provides item recommendations by first developing a model based on user preferences (by using data mining techniques), after that the system uses learning algorithms to look for habits according to the previously established model. Algorithms in this category take a probabilistic approach and envision the collaborative filtering process as computing the expected value of a user prediction, given users ratings on other items. The model building process is performed by machine learning algorithms such as Bayesian clustering, Bayesian network learning, and rule-based approaches.

Limitations

CF system should have information about data usage to be able to provide recommendations, thus it is not effective for new data, as such data cannot be recommended to any user because system cannot classify that data (if the data is not manually classified).

If the chosen approach is memory-based and modeling phase is performed in real-time then the system faces scalability limitations. As the numbers of users and items increase, this approach may lead to unacceptable latency for calculation of recommendations or creation of dynamic content during user interaction [32]. Workload sharing and data partitioning approaches can be used to scale such systems but they require the availability of vast processing capabilities (e.g., Hadoop [14] partitioning).

Another limitation is about the density of each user record that decreases with the increase number of items in the system; as the number of items increases, the likelihood of a significant overlap of used and/or rated items among pairs decreases resulting in less reliable computed correlations.

[14]http://hadoop.apache.org

3.3.2 Content-based Filtering

Content-based filtering (CBF) uses the content of items (set of features) that were rated by the user to reveal user preferences. It thus forms the user profile which is a structured representation of user interests and an essential component to estimate rate of interest for new items. In other words, the user profile represents the content descriptions of items that were used by user in the past. Items that are close (or the closest) to the user profile are considered as possibly interesting to the user (i.e., content-based filtering uses item features to determine similarity between rated and unrated items) and provided to the user as a recommendation. The coefficient of similarity between the user profile and the chosen item is considered as a likeliness-measure of user interest in a certain item.

Contrary to CF, CBF creates the user profiles based only on ratings that are provided by the particular user, without consideration of the ratings provided by other users. Knowing something about the content makes it possible to provide a rating estimate on new items (that were not rated by other users).

It is important to have a significant number of features of different types when describing items. There are several types of sources that can be used to discover features: i) the item itself (e.g., features that could be encoded into the items name, system attributes like time creation and item owner, etc.); ii) item description (e.g., metadata from an information system); iii) users (e.g., folksonomy, where users can assign some keywords or tags that are associated with items).

One general approach in item representation is the keyword-based Vector Space Model (VSM) with TF-IDF (Term Frequency - Inverse Document Frequency) weighting. Originally this approach was introduced as a text document representation (set of documents $\mathbb{D} = \{d_1, d_2, \ldots, d_M\}$) by a set of terms ($\mathbb{T} = \{t_1, t_2, \ldots, t_N\}$). Thus document d_m is represented by a N-dimensional vector: $d_m = \{w_{1,m}, w_{2,m}, \ldots, w_{N,m}\}$,

where $w_{n,m}$ is the weight for term t_n in the m^{th} document [30]. The term weighting scheme is TF-IDF, which calculates the frequency of each term in the document and relates it to the frequency of the same term in all of the documents, thus representing how relevant each and every term is to the current document.

$$TFIDF(t_n, d_m) = TF(t_n, d_m) \times \log\left(\frac{|D|}{|D(t_n)|}\right) \tag{3.7}$$

where $D(t_n)$ is the set of documents that contain term t_n at least once.

$$TF(t_n, d_m) = \frac{freq(t_n, d_m)}{max_z(freq(t_z, d_m))} \tag{3.8}$$

where $freq(t_n, d_m)$ is the frequency of term t_n for document d_m and $max_z(freq)$ is the maximum frequency over the frequencies $freq(t_z, d_m)$ of all terms t_z that are in document d_m. To bring all weights into the interval $[0, 1]$ the following normalization is applied:

$$w_{n,m} = \frac{TFIDF(t_n, d_m)}{\sqrt{\sum_{z=1}^{|T|} TFIDF^2(t_z, d_m)}} \tag{3.9}$$

Estimation of rate of interest for new items is based on calculating a similarity measure to determine the closeness between items. The most widely used method is cosine similarity (see the previous section for possible methods to calculate similarity).

The problem of learning user profiles can be also described as a binary categorization task, where each new item can be classified as either interesting to the user or not ($C = \{c^+, c^-\}$). The most popular learning algorithms in content-based filtering are decision tree based algorithms, nearest neighbor based algorithms, Rocchio's algorithm (based on relevance feedback), linear classifier based algorithms, and Naive Bayes classifier based algorithms (probabilistic methods).

Bayesian classifiers within a probabilistic framework can be used for solving classification problems; they utilize the definition of conditional probability and Bayes theorem. The Naive Bayes classifier is defined as:

$$P(c \mid i) = \frac{P(c)P(i \mid c)}{P(i)} \tag{3.10}$$

where $P(c \mid i)$ is a posteriori probability of item i belonging to class c, $P(c)$ is a probability of class c (a priori probability), $P(i \mid c)$ is a probability of observing item i given class c, and $P(i)$ is the probability of item i. To classify the item, the class with the highest probability is chosen:

$$c = argmax_{c_j} \frac{P(c_j)P(i \mid c_j)}{P(i)} \tag{3.11}$$

Limitations

CBF also has some disadvantages. Its performance is limited by the number of features that describe items. In CBF, it is not possible to discover new items that do not share features with any of the items in the user profile. Furthermore, the system should have a sufficiently high number of ratings from the user to provide accurate recommendations (i.e., it is not a good approach for new users). In general, CBF suffers from the problems of *limited content analysis* (i.e., the content of an item might be insufficient to determine its quality) and *over-specialization*.

3.3.3 Evaluation

The effectiveness (i.e., the quality) of a recommender system can be evaluated by calculating corresponding coefficients that show the rate of *True Positive* and/or

Negative recommendations towards *False Positive* and/or *Negative* (i.e., compare recommendations that were actually followed with that were ignored).

- True Positives (TP) - the number of instances classified as belonging to a defined class that indeed belong to that class;
- True Negatives (TN) - the number of instances classified as not belonging to a defined class that indeed do not belong to that class;
- False Positives (FP) - the number of instances classified as belonging to a defined class but which in reality do not belong to that class;
- False Negatives (FN) - number of instances not classified as belonging to a defined class but which in reality do belong to that class.

Precision is the fraction of retrieved instances that are relevant.

$$precision = \frac{TP}{TP + FP} \tag{3.12}$$

Recall is the fraction of relevant instances that are retrieved.

$$recall = \frac{TP}{TP + FN} \tag{3.13}$$

Using a probabilistic view, precision may be defined as the probability that an object is relevant given that it is returned by the system, while the recall is the probability that a relevant object is returned [33]. Using precision and recall, the frequently used, so-called $F1$-score is defined as:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} = \frac{2TP}{2TP + FN + FP} \tag{3.14}$$

According to [34] the *Matthews Correlation* (MC) is also well suited as a metric. MC is based on *markedness* and *informedness*.

Markedness combines precision and inverse precision into a single measure and expresses how marked the classifications of a recommender are in comparison to chance:

$$\begin{aligned} markedness &= precision + inversePrecision - 1 = \\ &= \frac{TP}{TP+FP} + \frac{TN}{TN+FN} - 1 \end{aligned} \tag{3.15}$$

Informedness combines recall and inverse recall into a single measure and expresses how informed the classifications of a recommender are in comparison to chance:

$$\begin{aligned} informedness &= recall + inverseRecall - 1 = \\ &= \frac{TP}{TP+FN} + \frac{TN}{TN+FP} - 1 \end{aligned} \tag{3.16}$$

Both markedness and informedness return values in the range $[-1, 1]$. The MC then combines the markedness and informedness measures into a single metric by calculating their geometric mean (the $\pm$ notation in the formula signifies that the sign depend on the signs of the markedness and informedness scores):

$$\begin{aligned} MC &= \frac{TP \cdot TN + FP \cdot FN}{\sqrt{(TP+FN)(FP+TN)(TP+FP)(FN+TN)}} = \\ &= \pm\sqrt{markedness \cdot informedness} \end{aligned} \tag{3.17}$$

3.4 Summary

In this chapter we have learned about techniques and approaches that fit our model in revealing relationships between objects or ordered sets of objects, techniques

that provide the description of how elements are correlated to each other based on specific characteristics, and how this information can be used in recommender systems. Our focus will remain on recommender systems, as a machine learning technique in the oncoming investigation of user preferences and interests (based on predictions of corresponding ratings).

CHAPTER 4

THE VIABILITY OF A RECOMMENDER SYSTEM

In this chapter we will be investigating whether or not data coming from PanDA has hidden patterns in it that could indicate similarities between users and thus indicate that a recommender system could be to the benefit of users. Thus, given the per user data usage of datasets (i.e., items) in PanDA, our task is to find correlations among users and items. We represent data in a transactional form, i.e., each sequence of transactions is associated with a user, and represents the sequence of itemsets that the particular user used during the transaction time window.

The first step to reach our goal is to find associations within items. Such associations will state that there is a relation between two items if the usage of the first item tends to indicate the usage of the second item (regardless of relations between users). Having associated items will provide some indication that there are frequent sequences of itemsets. Establishing then the presence of sequential patterns (relying on these associations found) will provide confirmation that relationships between user data usages may exist. More precisely, sequential patterns for which the support is greater than an analyst-specified minimum support will be considered frequent and thus show that there are associations between users' data usages.

Strong correlations between data usages will provide indication that the future employment of data mining techniques for classifying user preferences and to predict user's future activities (data usage) will likely lead to a usable recommender system for PanDA users.

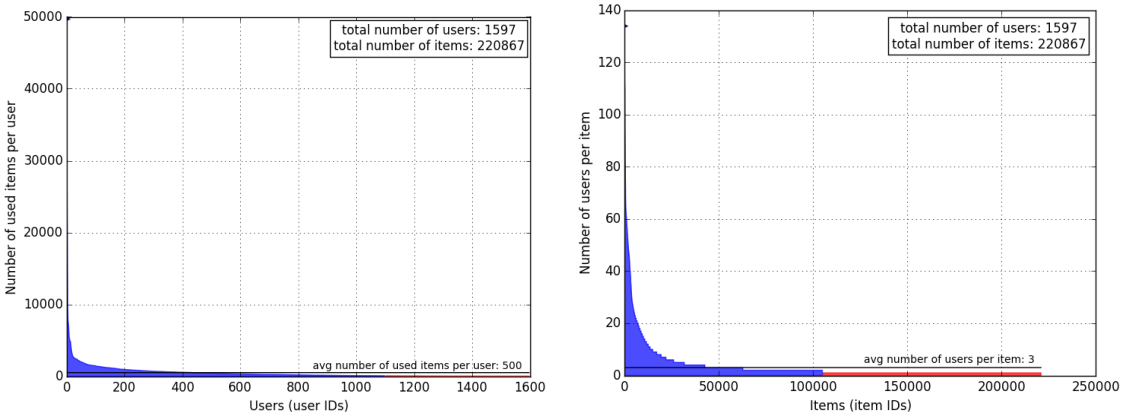
The following analysis requires an interaction process with users that have submitted analysis jobs to PanDA. Based on user activity at PanDA (time when jobs were submitted, how many jobs were finished successfully, what type of data were used as input, etc.) users are provided with the list of items with corresponding ratings (indicating the possibility that items might be in the user's interest area).

The result of this investigation should be an evidence (or lack of evidence) of the existence of correlation between user's data needs. Since for this investigation we do not need to use live data, information from "archive" tables of the PanDA database was used. To be able to cover different states of users interests (i.e., increase/decrease of interest in certain type of data) it would help to reduce or eliminate the influence of other intelligent systems. Fortunately, we were able to identify PanDA data for the entire year of 2011 when certain intelligent PanDA subsystems (such as PD2P) were not in full production yet. As it was mentioned earlier, PanDA collects information about users, their jobs (as records with specifications as parameters), and data that is used as the input during the processing of these jobs. The corresponding historical year-2011 data contains about 220M records, with $1,597$ relevant users (out of $1,814$ in total) and $220,867$ relevant items (out of $380,111$ in total). (Relevant users are defined as users who had analysis jobs that terminated successfully; relevant items have a similar definition related to success and parsability.)

Figures 4.1a and 4.1b show how many items were used by each user (indexed by consecutive user ids) and how many users had requested certain items (indexed by a consecutive item id) accordingly. The data shows that:

- 95.6% ($1,526$ users) of all relevant users had requested at least two distinct items;
- 68.9% of relevant users ($1,100$ users) had requested more than 10% of the average number of requested items (Figure 4.1a, blue color);

- 47.6% of relevant items (105, 097 items) were requested by at least two distinct users (Figure 4.1b, blue color).



(a) Number of items per user

(b) Number of users per item

Figure 4.1: Data usage histograms for users and items

In this study we ignore items that have only been used by one user; however, in a future real recommender system such items may be considered if they show similarities with other, more popular, items.

The average number of (distinct) items used per user is 500 with a standard deviation of 1, 758. This large standard deviation indicates strong skew; 39 users heavily exceed the average (by more than a standard deviation). As users with such large number of items used would overwhelm with their recommendations, we choose to remove them from this analysis leaving us with 1, 558 users. (Indeed keeping these users in our analysis would show a much larger number of frequent data sets but may be misleading as to how useful such recommendations would be. Furthermore, our analysis of these users indicates that many of them either represented artificial job requesting agents or used each item only once. In general, data of such users would introduce too much noise into the analysis.)

4.1 Data Modeling and Representation

One of the sub-goals of a data mining application is to create a set of data models, that can be used as input to a variety machine learning algorithms for pattern discovery. In our case such data models would represent the interests and activities of all users thus defining profiles. The output of our data mining application, i.e., the discovered patterns, could then be used for predicting future interests of users [35].

We elect to represent the data in a transactional form. Thus, we define the set of transactions $\mathbb{T} = \{t_1, t_2, \ldots, t_M\}$, where t_m is a transaction that aggregates objects from the following sets: set of users $\mathbb{U} = \{u_1, u_2, \ldots, u_K\}$ and set of items (used data) $\mathbb{I} = \{i_1, i_2, \ldots, i_N\}$. Each transaction t_m is characterized by its owner, set of used data, starting time, and duration. Thus $t_m = \{eid, u_k, I(t_m) \mid u_k \in \mathbb{U},\ I(t_m) \subseteq \mathbb{I}\}$, where $eid(eventid)$ is a transaction start time, u_k is transaction's owner, $I(t_m)$ is a nonempty set of items that were used by user u_k within the transaction t_m (i.e., the itemset or k-itemset means that it contains k items). Since a transaction is supposed to describe what items are requested during a certain time window by a user, we restrict the duration of transactions to a common transaction time window Δt. (At the creation of the transaction database, if an activity is longer than Δt then it gets split into several Δt long transactions.)

Transactions can then be grouped by their users; inside each such group, transactions are ordered by their start time (eid), thus representing a data sequence s_k for user u_k. The set of all data sequences then is: $\mathbb{S} = \{s_1, s_2, \ldots, s_K\}$, where $s_k = \{u_k, T(u_k) \mid u_k \in \mathbb{U},\ T(u_k) \subseteq \mathbb{T}\}$, and $T(u_k)$ is an ordered subset of transactions for user u_k.

The cumulative set of user features and preferences can be represented as a degree of interest of a user to a specified item; this cumulative set can be seen as the user profile. The set of user profiles is thus $\mathbb{UP} = \{up_1, up_2, \ldots, up_K\}$, where up_k is a

user-specific function that calculates the rate of interest of user u_k to an item i_n, i.e., $interest\ degree(u_k, i_n) = up_k(i_n)$. The interest degree (rate of interest) value might be represented as a binary value (interested or not interested), or as numeric value (e.g., actual ratio).

4.2 Transaction Time Window Estimation

To get a reasonable evaluation by the item association analysis, data usage should be investigated first to find a good estimate for the transaction time window (TTW). Thus potential recommendations of items, which were not used by users before, would be simulated (these recommendations will be stored and cross referenced against real item use). The recommendations would be based on the similarity between users (i.e., similarity in data usage).

4.2.1 Problem Definition

For every user $u_k \in \mathbb{U}$, we maintain three associated subsets:

- Used items $UI(u_k) = \{i_j : [(tu_{kj}, n_{kj})] | i_j \in \mathbb{I}\}$, where tu_{kj} is the date when item i_j was used by user u_k; and n_{kj} is the number of times (i.e., number of jobs) the corresponding item was used during that day. (This implies, that a specific item may have several (tu_{kj}, n_{kj}) pairs associated with it.)
- Recommended items $RI(u_k) = \{i_j : \{u_m : [(tr_{kmj}, sc_{km})] | u_m \in SU(u_k)\} | i_j \in \mathbb{I}\}$, where tr_{kmj} is a timestamp, time when user u_m had used item i_j; and sc_{km} is a similarity coefficient between users u_k and u_m. Thus this set contains items that user u_k may receive as recommended based on similar interests.
- Similar users $SU(u_k) = \{u_m : [(ts_{km}, sc_{km})] | u_m \in \mathbb{U}\}$, where ts_{km} is a timestamp, time when similarity between two users was evaluated by the analysis

system; and sc_{km} is a similarity coefficient between users u_k and u_m. (When sc_{km} is updated then ts_{km} is also updates.)

The goal of our simulation based analysis is to find an average time difference between potential recommendation and actual data usage. Again, this will help us determine a proper transaction time window needed for the sequential pattern mining.

4.2.2 Evaluation

As described previously the processing time period for used items is set to calendar days. For every user during each processing period the following actions have been applied:

1. Maintain $UI(u_k)$: the list of objects for used items are created and inserted into $UI(u_k)$, if one of these items was recommended earlier (i.e., the same item was already in $RI(u_k)$) then move corresponding item from recommended items to the certain used item object (a list that is maintained in the simulation);
2. Maintain $SU(u_k)$: maintain the sets of similar users, i.e., evaluate every user-pair and create corresponding objects at $SU(u_k)$ with the Jaccard index (Formula 4.1) as the similarity coefficient, if this coefficient exceeds a threshold value;

$$jsc(u_k, u_m) = \frac{|UI(u_k) \cap UI(u_m)|}{|UI(u_k) \cup UI(u_m)|} = (\frac{num\ common\ used\ items}{num\ total\ used\ items}) \qquad (4.1)$$

3. Maintain $RI(u_k)$: create new recommendations by inserting the used items of just established similar users into $RI(u_k)$.

Figure 4.2 shows the per day average (per user) number of items used (and its standard deviation), the average number of recommended items in $RI(u_k)$, and the

true positive recommendations (related to the certain used object list size). PanDA does not currently have a recommender system; thus the data based on which our simulation analysis is performed was not influenced by recommendation induced item usages. As we are interested in establishing possible relationships between the interests of different users to see if a recommender system would add value to their work, in this analysis we are not going to consider false positives and false negatives among our data sets.

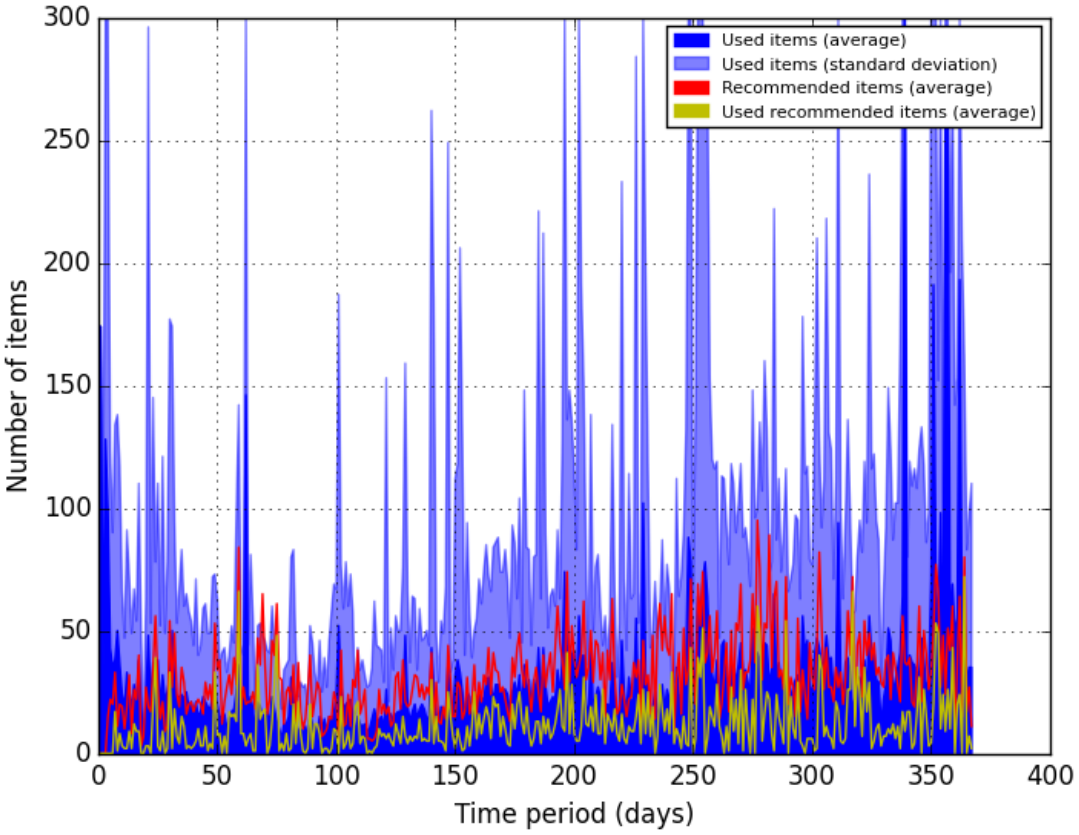


Figure 4.2: Number of items (used and recommended) per day during the analysis period (y-axis is truncated to 300 items)

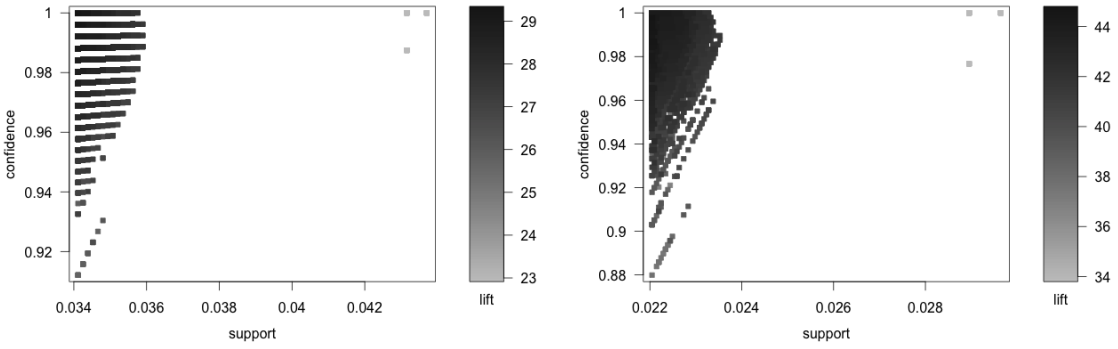
Further analysis revealed that the average time difference between a potential recommendation and actual data use among the true positive simulated recommendations is about 28.7 days with a standard deviation of about 33.5 days. For the sake of simplicity, we elected to use a transaction time window of 30 days for the following analysis.

4.3 Association Rule Mining

For most of our association analysis we used R, a language and environment for statistical computing and graphics. Association rules were generated with *arules* [36] (a computational R library for mining association rules and frequent itemsets) by using the Apriori algorithm. Table 4.1 shows the results for applying the arules tool; data in columns headed with italics were used as inputs to arules. We varied the minimum support to show its impact on the number of associated items and rules, and elected to insert the more meaningful values into the table. Figure 4.3 shows plots of how indeed the minimum support influences the number of associations and rules.

Table 4.1: Numbers for association rules (generated with R library *arules*)

TTW, days	Number of transactions established	*Minimum support*	*Minimum confidence*	Number of associated items found	Number of associated rules created
30	7,299	0.036	0.5	23	28
		0.035	0.5	71	43,446
		0.034	0.5	150	15,652,534
		0.0335	0.5	166	39,144,905
5	20,276	0.023	0.5	55	1,629
		0.022	0.5	141	9,434,742
		0.0215	0.5	161	39,563,180



(a) (b)

Figure 4.3: Minimum support influence scatter plots (a) for 7,299 transactions (TTW of 30 days); (b) for 20,276 transactions (TTW of 5 days)

With a transaction time window of 30 days most associated items do not have a support greater than 0.036 (3.6% of transactions contain associated items); with smaller transaction time windows this upper threshold for support becomes even smaller. The results show the existence of associations for 30 day windows at a valid minimum support.

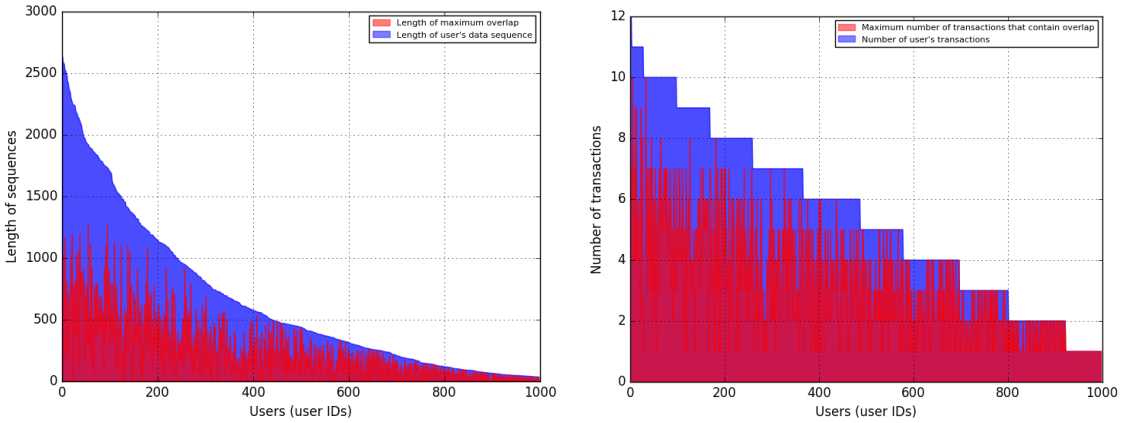
4.4 Frequent Sequences in Data

The association analysis in Section 4.3 confirmed that certain subsets of used items are correlated as far as user interests are considered; however this is not enough to conclude that sequences of used items are correlated as well. As described earlier, sequential patterns between related users would show to what extent their usage activities overlap. Users' data usage activities are represented as data sequences with the previously determined transaction time window of 30 days (the maximum number of transactions per user is 12 in the extracted PanDA data).

There are several implementations of sequential pattern mining algorithms; for our purposes the most applicable representatives were identified as: i) the R package *arulesSequences* - mining frequent sequential patterns with the cSPADE algorithm [37]; and ii) SPMF (Sequential Pattern Mining Framework) - an open-source data mining library written in Java, specialized in pattern mining [38]. Both tools worked and showed good results only with certain portions of our data, but not with the entire data set. The number of sequences in conjunction with their sheer sizes were causing frequent crashes in the above tools. Thus we took it upon us to create our own sequential data mining tool, specifically tailored to handle the intricacies due to the dimensions of our data. As described before, our algorithm of choice for sequential pattern mining is the Apriori-based - SPADE [28]. In addition to an implementation of SPADE, we extended it with CMAP (Co-occurrence MAP) [39] and bit sequence representation; this tailoring of SPADE was done to increase the computational efficiency. Our implementation in Python is available at [40].

Our custom SPADE tool was applied to every pair of users to detect the degree of their correlation. The average sequence length (the average sum of lengths of transactions) in our transactional database is about 700. Sequential pattern mining revealed that for 998 users (64% of relevant users) the sequences share a significant portion of the timeline of used items with at least one other user, i.e., there is a sequence pattern (for every pair of sequences) that is a subsequence for each of the paired sequences. More precisely, we found that the average length of a sequential pattern is 43.58% of the lengths of the original sequences (with a standard deviation of 27.25%). To avoid biases coming from short, completely overlapping sequences, we removed the outliers and only considered sequences with lengths greater than 5% of the average length. (The number of users with sequence lengths less than 5% of the average length is 269 or 16.8% out of the total number of relevant users.) Figure 4.4

shows histograms of the lengths of users' data usage sequences with the corresponding maximum overlap. This significant average length of sequential patterns (43.58%) indicates a strong correlation between users' data needs and thus validates our belief that a recommender system would enable users to find interesting data more readily and rapidly for their experiments.



(a) Length of sequences per user

(b) Number of transactions per user

Figure 4.4: Maximum data usage overlap per user

4.5 Summary

In this chapter we investigated historical PanDA data; we have implemented a custom version of the SPADE sequential pattern mining algorithm with extensions that accommodate the dimensionality of said data. Deeper analysis, that included considerations for relationships between items in relation to users, presented correlations between users based on items relations. Indeed we found that data usage activity shows about 44% overlap for 64% of all relevant users. We also found that another about 17% of all relevant users had overlapping, but not significant data

usage correlations. These findings indicate a strong correlation between users' data needs, validating our belief that a recommender system would enable users to find interesting data more readily and rapidly for their experiments. Indeed our results have helped us raise awareness of the potential benefits of a recommender system to PanDA.

CHAPTER 5

DATA WATCHER DESIGN

The previous chapter confirmed our belief that there are correlations between users and user data needs in PanDA, and that almost every user activity contains usage patterns that could be revealed. The ability of recommender systems to explore user activities in the past and discover that hidden relationships between users' data needs lead us to use these methods in designing Data Watcher (DW), a system that aims at helping users in identifying and tracking data. In this chapter we will provide a detailed explanation of Data Watcher, the system's design, and core algorithmic components of the system.

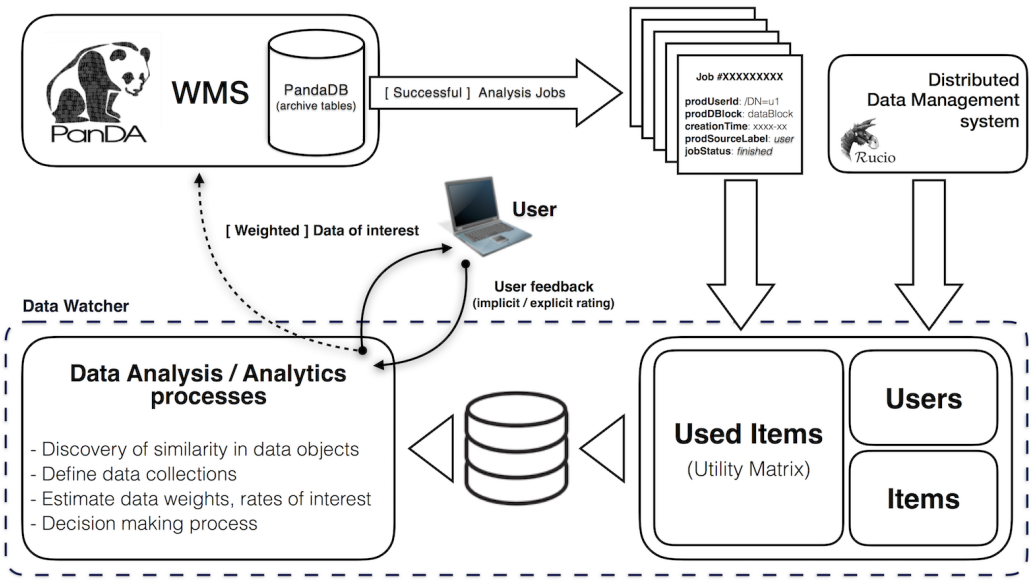


Figure 5.1: Data Watcher workflow organization

Data Watcher was designed to be able to estimate the degree of dependency and demand between data objects in the PanDA system (e.g., users, and items that represent input data for users' processing tasks), and to utilize this knowledge to provide corresponding ratings for particular objects. At the heart of DW are algorithms based on both CF and CBF approaches; users can access DW using a web-interface to monitor data ratings and corresponding coefficients (e.g., data popularity, the degree of user interest, etc.). Figure 5.1 illustrates the way DW interacts with PanDA and Rucio systems.

5.1 System Workflow

The Data Watcher system contains three essential components: i) database; ii) web-application; and iii) scripts and code for offline data processing (the core of the system). Figure 5.2 shows a UML class diagram of DW with the most substantial class attributes. Here, we are going to provide a detailed description to accompany Figure 5.2. *Account* and *Item* are the primary classes of DW; the other classes aid the primary classes by providing further descriptions to the primary classes or represent some relations between them.

Class *Account* (and its sub-class *UserAccount*) represents PanDA users (i.e., the abstraction of a user); this can be: i) an explicit individual user; ii) a service account (i.e., accounts related to but external to PanDA, e.g., a separate system that is used to interface with PanDA to submit jobs on behalf of the user); or iii) group account (e.g., physics group account). *Accounts* are mapped to the owner of corresponding PanDA jobs. (*AdminAccount* is a regular user with extra permissions to manage system data). Class *Interest* defines keywords that represent the *Accounts*' interests; they are mapped to some item attributes that describe item content. Class *AccountLabel* describes unique *Account* identities (e.g., X509 certificate distinguished

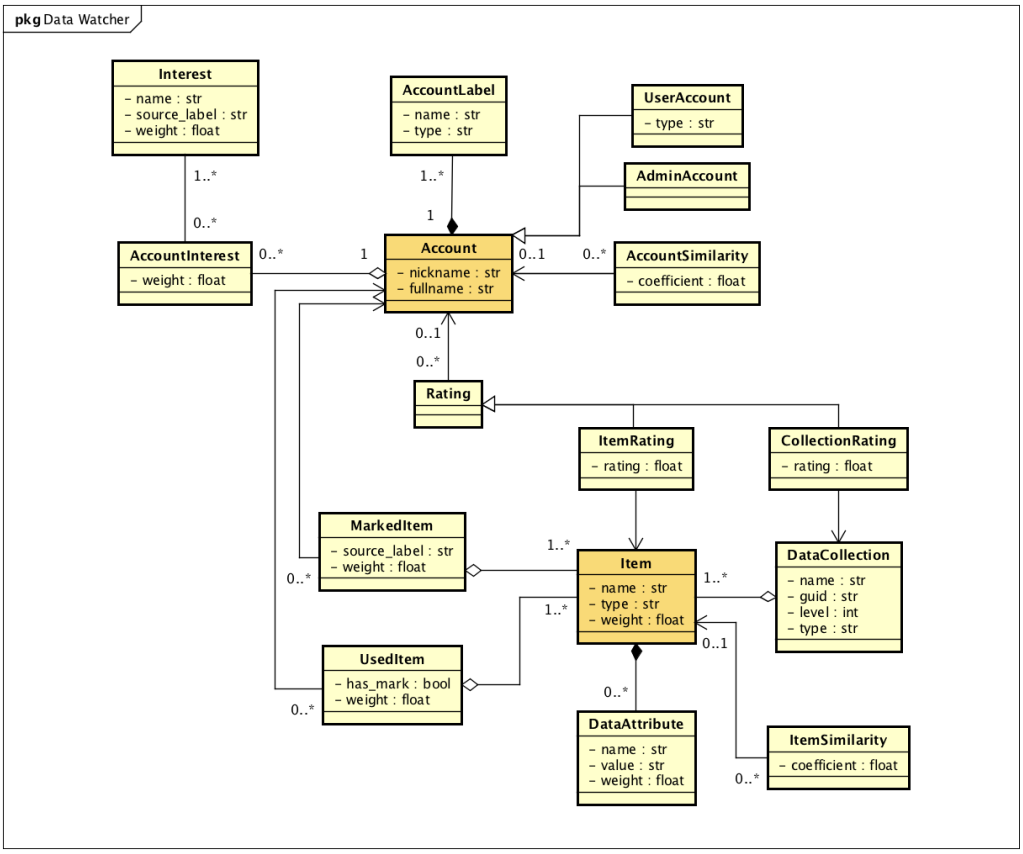


Figure 5.2: Data Watcher class diagram

name, e-mail, unique alias). Class *AccountSimilarity* represents a relation between two *Accounts* using a similarity measure.

Class *Item* is a unit of input data in PanDA jobs. Similarly to class *AccountSimilarity*, *ItemSimilarity* represents a relation between two *Items* respectively with a similarity measure. Class *DataAttribute* characterizes the corresponding *Item* (describes the *Item* itself or its content). Class *DataCollection* represents a collection of *Items* that are grouped because of a particular rule or corresponding distance measure between entities. Class *UsedItem* captures the list of *Items* that were used by a particular *Account*. Class *MarkedItem* is used to maintain the list of *Items* that were

marked as possibly interesting for a corresponding *Account*. Class *Rating* represents the *Account*'s feedback as a rating value to the corresponding *Item* or *DataCollection*.

Users interact with Data Watcher using a web-interface. The interactions of the system with the user and the modules of the system are shown as a UML communication diagram in Figure 5.3.

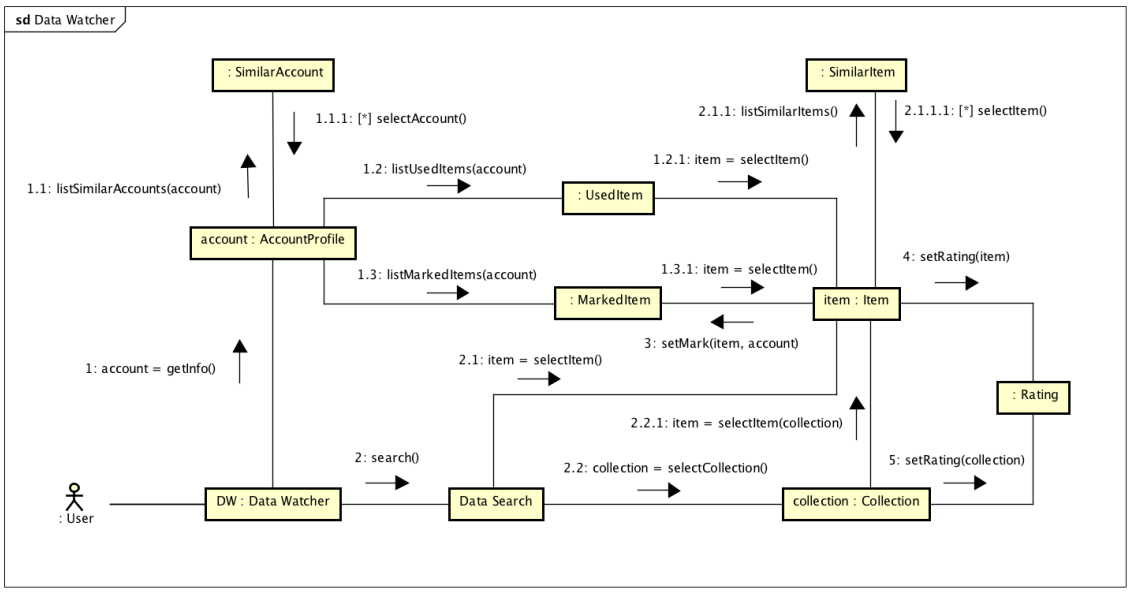


Figure 5.3: Data Watcher communication diagram

DW's web interface provides the means for the users to discover new data (items) and explore revealed relationships between users and between their interests. Every user is associated with a corresponding profile (path 1) that supplies such information as:

- the list of similar users (path $1 \rightarrow 1.1 \rightarrow 1.1.1$), i.e., users that have the largest intersection of interest with the current user;
- the list of used items (path $1 \rightarrow 1.2 \rightarrow 1.2.1$), i.e., items that were used by the user as input data for a processing task;

- the list of marked items (path 1 → 1.3 → 1.3.1), i.e., items that DW marked for the user as possibly interesting (fitting the user's interest area).

Paths 2 → 2.1, and 2 → 2.2 → 2.2.1 will lead the user to the description of items to which they want to find similar items (path continue from "item:Item" 2.1.1 → 2.1.1.1). Following path 1.2.1 → 1.2 can reveal to the user who has used the particular item.

Users can interact with the core algorithms by:

- providing personal recommendation (*source_label="user"*) of chosen items to other users (by taking path 3);
- providing explicit rating for a particular item (path 4); and/or
- providing explicit rating for a particular collection of items (path 5).

As described before, at the heart of DW is a recommender subsystem; in the next section we are going to dwell into the internal workings of this subsystem.

5.2 The Recommender Subsystem

In order for Data Watcher to actually work, it needs to rely on a custom built recommender system. In this section we are going to detail the approach that was used. Here, we may use a convention to represent matrices as functions, where input parameters correspond to indices, e.g., $M = [m_{i,j}] \Leftrightarrow M(i,j) = m_{ij}$, we believe that such representation will help the reader to better follow relations between objects.

5.2.1 Data Representation

There are two main sets of data objects: set of users $\mathbb{U} = \{u_1, u_2, \ldots, u_K\}$ and set of items (input data for user processing tasks) $\mathbb{I} = \{i_1, i_2, \ldots, i_N\}$. Each item can be represented by a set of attributes (features) $\mathbb{A} = \{a_1, a_2, \ldots, a_M\}$. Thereby, each

user can be represented by a user profile that is a vector of weights that are associated with the corresponding attributes (same index): $UP(u_k) = \{w_{1,k}, w_{2,k}, \ldots, w_{M,k}\}$.

5.2.1.1 Utility Matrix

The Utility Matrix (user-item matrix) is an essential part of the core recommender subsystem; it represents connections between users and items. An element in this matrix is a quantitative representation of such relation, i.e., a rating (or interest degree that shows how much user u_k is interested in item i_n, thus $rating_{k,n}$).

$$UM(u_k, i_n) = [rating_{k,n}] \tag{5.1}$$

Initially, the Utility Matrix ($UM_{initial}$) contains the number of (successful) analysis jobs that were created by user u_k with the input data represented as item i_n.

$$UM_{initial}(u_k, i_n) = num\_jobs_k(i_n) \tag{5.2}$$

There are two facts that should be considered here in the context of PanDA: i) each item (i.e., data unit) may contain a different number of files; ii) the number of (successful) jobs usually is proportional to the number of used/requested files in the input data. Thus, in order for the numbers to be comparable (as the distribution not the absolute number is of importance) the utility matrix should be *normalized*.

The first step of the normalization is to calculate for each user the ratio between the number of user jobs with a particular item (input data) and the average number of jobs that used the same item:

$$avg\_num\_jobs(i_n) = \frac{\sum_{u_z \in U(i_n)} UM_{initial}(u_z, i_n)}{|U(i_n)|} \tag{5.3}$$

$$UM_{shifted}(u_k, i_n) = \frac{1}{avg\_num\_jobs(i_n)} \times UM_{initial}(u_k, i_n) \tag{5.4}$$

where $U(i_n)$ is a set of users who had requested item i_n as an input data for their jobs. Hence, the first step of normalization is to calculate the ratio between the number of jobs and the average number of jobs per item. $UM_{shifted}(u_k, i_n)$ will show user interests according to the average data usage among users who used that data; in the future we may refer to the scalars in $UM_{shifted}(u_k, i_n)$ as the *variation of number of jobs* from the average for item per user.

The second step of the normalization is to normalize elements among items per user, i.e., calculate coefficients (interest degree) for items per user.

$$UM_{normalized}(u_k, i_n) = \frac{1}{max_z(UM_{shifted}(u_k, i_z))} \times UM_{shifted}(u_k, i_n) \tag{5.5}$$

This final normalized matrix thus contains interest degrees for items for a particular user that considers user interest for particular data among other users that used that data, and then utilizes that ratio to reveal user preferences among this particular user's used data. (In our case ratings are scalars, i.e., they belong to the numerical class of ratings.)

5.2.1.2 Data Popularity

Ratings estimation for items that are potential recommendations relies on the popularity of data as one of the essential parameters. Data popularity is calculated for the defined period of time and includes the *aging factor*, that is used to be able to differentiate data created before the defined time period and during the time period

(i.e., to emphasize the difference between "old" and "new" data that have the same number of accesses).

$$popularity(i_j, \Delta t) = \frac{|U(i_j, \Delta t)|}{|U(\Delta t)|} \times aging\_factor, \quad \Delta t = [t_0, t_1]$$
$$aging\_factor = \begin{cases} 1, & \text{if } creation\_date(i_j) \in \Delta t \\ \frac{1}{t_0 - creation\_date(i_j)+1}, & \text{if } creation\_date(i_j) \notin \Delta t \end{cases} \tag{5.6}$$

5.2.2 Similarity Estimation

Collaborative filtering (described earlier and applied in an oncoming section) is based on metrics describing similarities between users; because one approach of calculating user similarity comes from the content-based filtering field, similarity estimation will be presented here in a separate sub-section.

There are two (chosen) ways to estimate similarity between pairs of users: i) compare user activities (i.e., compare how many times users had requested the same data for the defined period of time); and ii) compare user preferences in certain item attributes (i.e., compare how many times common attributes with corresponding weights appeared in requested data for the defined period of time).

The basic representation of user activity is a set of items that were used (or requested) by the user during a specified period of time. (For further improvements, the system will use the earlier presented description of the activity, i.e., an ordered set of transactions, where each transaction is characterized by the user, the set of items that were used/requested by particular user, the transaction start time and its duration.) This information will help to estimate coefficients of similarity between pairs of users in terms of used data; however such information does not reflect the

magnitude of user interest in each of the items. Thus we extended the Jaccard index with a correlation coefficient (that could be viewed as an opposite to the distance between corresponding *variances of number of jobs*, i.e., $UM_{shifted}(u_z, i_n)$).

$$\begin{aligned} sim(u_k, u_p) &= \frac{\sum_{i_z \in I(u_k) \cap I(u_p)} d(u_k, u_p, i_z)}{|I(u_k) \cup I(u_p)|} \\ d(u_k, u_p, i_z) &= e^{-|UM_{shifted}(u_k, i_z) - UM_{shifted}(u_p, i_z)|} \end{aligned} \tag{5.7}$$

Similarity based on user preferences for attributes and items is calculated according to the cosine similarity measure. This approach considers the weights of attributes in user profiles (that are calculated by the TF-IDF technique and represent the significance of the corresponding attribute) and user interest degrees in data (elements of normalized utility matrix). Thus:

$$sim(u_k, u_p) = \frac{\sum_{a \in A} w_k(a) \cdot w_p(a)}{\sqrt{\sum_{a \in A} w_k^2(a)} \sqrt{\sum_{a \in A} w_p^2(a)}} \tag{5.8}$$

where A is the union of attributes from user profiles $UP(u_k)$ and $UP(u_p)$, and $w_z(a)$ is the corresponding weight for attribute a.

5.2.3 Collaborative Filtering

Collaborative filtering has been described in general in Section 3.3.1; this technique creates the possibility to consider the preferences of other users that are similar to a particular user. In DW, each user is placed in a group of similar users that is based on the similarity between users' activities and profiles. DW can then place the most popular items among users in the current user's similarity group into the user's recommendation list.

The major accent of this approach is on the valid estimation of similarity between users, thus we consider both of the following sources: pure data usage numbers and user preferences in used data content (described in previous section). The secondary task then is to select the most relevant items from the group of similar users. Thereby, DW groups data into collections and creates a hierarchy of collections in a tree structure, based on data origin. Table 5.1 shows an example for the tree of collections; every collection is represented as a tree node and items that belong to the collection with the highest level number are represented as leaves.

Table 5.1: Example of the tree of collections (without corresponding items)

Collection name	Level
AcerMC_Wt	0
mc11_7TeV.AcerMC_Wt	1
mc11_7TeV.AcerMC_Wt.AOD	2
mc11_7TeV.AcerMC_Wt.NTUP_SMWZ	2

Every *item* that is used by a user from the group of similar users is associated with a corresponding *collection*. Those collections that include items which were used by every user from the group are picked as a potential source for data recommendations. Items (from corresponding collections) with the highest cumulative metric based on popularity and user ratings are chosen for recommendations.

5.2.4 Content-based Filtering

Content-based filtering is based on explicitly provided item attributes (features) applied to user profiles. Item attributes in DW are features that were extracted from the item name (according dataset nomenclature in the ATLAS experiment): *projectName*, *dataType*, *configTag*, *datasetNumber*, *physicsShortName*, *productionStep*,

runNumber, *streamName*, *swReleaseComment*, *containerName*, *containerType*, *containerVersion*, *group*, *user*.

The user profile should contain information about item attributes that are interesting to the particular user, with corresponding degrees of interest (i.e., weights). In terms of the TF-IDF weighting scheme, weights for item attributes can be represented as AF-IIF (Attribute Frequency - Inverse Item Frequency) for collections of items or as AF-IUF (Attribute Frequency - Inverse User Frequency) for user profiles as collections of user preferred attributes. AF-IUF weights are used to show the importance of attribute (compared to other attributes in the user profile):

$$\begin{aligned} AFIUF(a_j, I(u_k)) &= AF(a_j, I(u_k)) \times \log\left(\frac{|U|}{|U(a_j)|}\right) \\ AF(a_j, I(u_k)) &= \frac{counter_j(I(u_k))}{max_z(counter_z(I(u_k)))} \end{aligned} \tag{5.9}$$

where $U(a_j)$ is a set of users who have attribute a_j in their profiles; $counter_j(I(u_k))$ is the frequency of attribute a_j for a set of items $I(u_k)$ used by user u_k; $max_z(counter)$ is the maximum frequency over the $counter_z(I(u_k))$ of all attributes a_z for user's used items $I(u_k)$.

Finally, the normalized attribute weights $w_{j,k}$ for attribute a_j for user u_k can be calculated as:

$$w_{j,k} = \frac{AFIUF(a_j, I(u_k))}{\sqrt{\sum_{z=1}^{|A|} AFIUF^2(a_z, I(u_k))}} \tag{5.10}$$

Content based filtering could be further improved by exploiting possible associations between attributes. We will further elaborate on this in the future work section of the final chapter.

5.3 Summary

In this chapter we described the core components of the Data Watcher system, the organization of its internal classes. We have also provided a communication schema that illustrates how the system assists users in the discovery of new data (based on similarity between users and data similarity). We then described the way data was processed and the methods that were applied to obtain the corresponding data ratings and weights (by highlighting the significant and specific components). In the following chapter we will discuss the applicability of Data Watcher and show preliminary results obtained.

CHAPTER 6

STUDYING DATA WATCHER

Data Watcher was designed as an independent subsystem in the PanDA ecosystem that provides an analytical information to users, as well as to PanDA Brokerage subsystem. The core components of the system are organized in such way that Data Watcher can easily interact with other WMS and DDMs. There is a specific component in DW, the data collector, that can be easily swapped out and tailored to specific WMS and DDM systems. (Currently the data collector uses PanDA and Rucio as information sources.) As a consequence DW is able to analyse data coming from systems that store information about connections between users and data, notwithstanding the high rate of new data production or that of the reprocessed versions of existing data. The current implementation of the component that is responsible for the collection of information from PanDA and Rucio effectively manages the organization of data and its representation, the organization of the data processing and the way it is distributed. Thus only the data collector part of DW depends on the information sources and how that information is transformed into corresponding objects inside DW.

This chapter will provide an insight into what is expected from DW by showing results that were gathered based on PanDA data. We will present some user-data statistics and provide preliminary results on the performance of the analytical processes. We will also provide an overview of how the user experience is enhanced and discuss potential future improvements and extensions.

6.1 Basic Statistics

Data Watcher started to collect information from the PanDA system in May of 2015. Once a day the DW collector component extracts the successfully finished analysis jobs from the archival table of the PanDA database (using the parameters *prodSourceLabel*=*"user"*, *jobStatus*=*"finished"*) and with the help of calls to the Rucio system (that contains users and data descriptions) transforms it into DW objects. At the time of this writing, the DW database contains about $1,700$ user objects, about $570,000$ item (dataset) objects, and about 1.3 million user-item records (i.e., elements that make up the the utility matrix, described in Section 5.2.1.1).

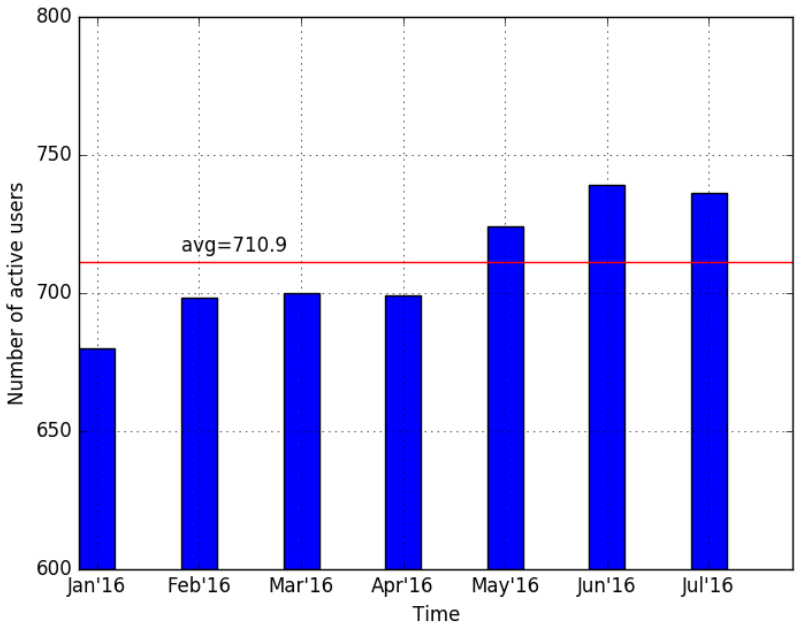


Figure 6.1: PanDA active users (that have successfully finished analysis jobs) per month

Figure 6.1 shows the numbers of users that were active in the PanDA system during each of the last seven months. Figure 6.2 shows the number of active user-item records in Data Watcher per month (for the same duration); this shows how many

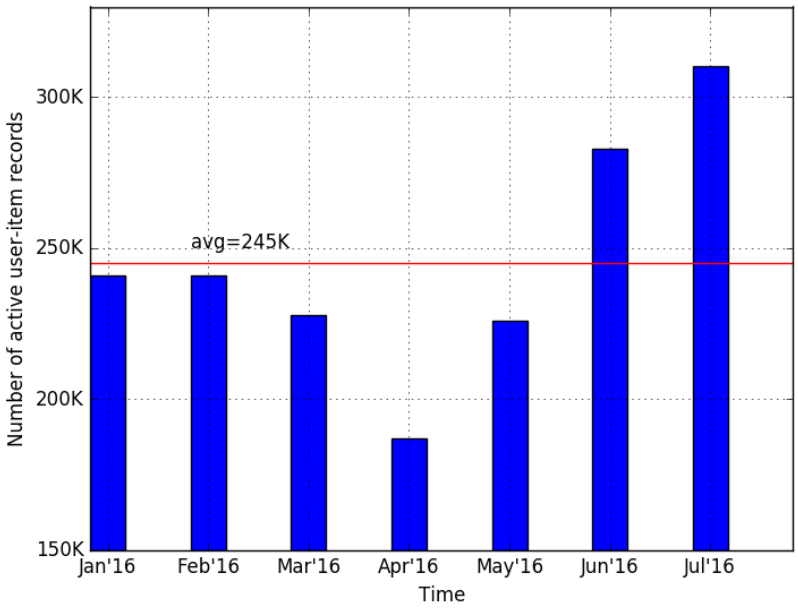


Figure 6.2: Data Watcher unique user-item pairs per month

unique user-item pairs were used per month (it does not represent how many times users have used the same data, it merely provides an estimate on the size of the utility matrix in each month).

6.2 Preliminary Results

In order to analyze the efficiency of the applied recommender system, we will consider the two core algorithmic components separately (i.e., the collaborative and the content-based part) to evaluate to what extent they contribute to their combined result.

Collaborative filtering is considered as the more essential part of DW, due to its possibilities to discover data that will benefit users. This approach considers data that is in the interest areas of users who are similar to the user for whom DW produces the recommendations. Thus, DW creates a new connection between the user and data with a corresponding likeliness score (i.e., probability of user interest

in given data). With only the collaborative filtering component turned on, Data Watcher shows on average of around 15% of successful recommendations (the ratio between True Positive and False Positive values), with an average time lag between the recommendation and the actual data usage of about 22 days. The average number of distinct users who received recommendations is about 45 per month. Figure 6.3 shows a comparison chart of recommendations provided versus followed (i.e., items that were marked for users, and marked items the were actually used later by users).

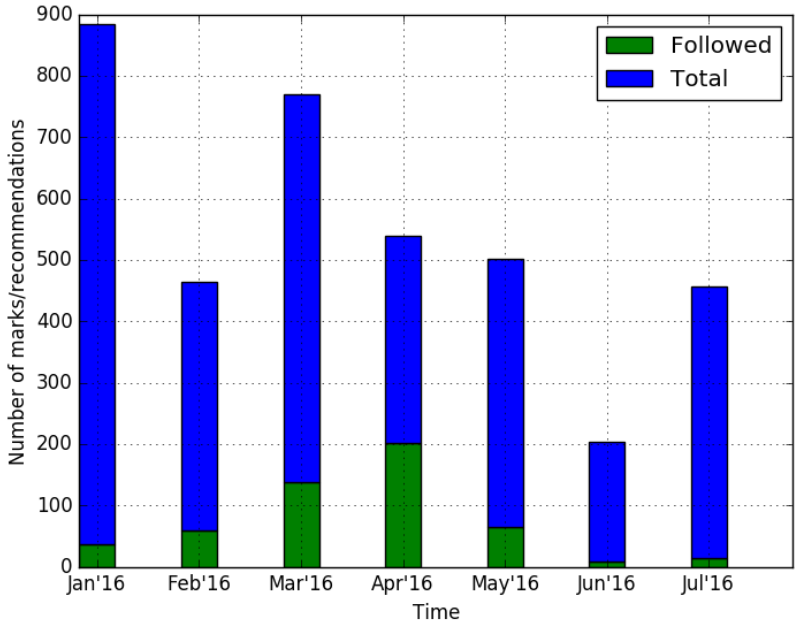


Figure 6.3: Comparison of the provided and followed recommendations per month (using only the collaborative filtering component)

Using only the content-based filtering component in DW, such parameters as: weighted data attributes, similarity between items based on content (i.e., set of attributes with pattern-based matching) are included. Since some sets of data are rather poorly described with the filename provided attributes (i.e., partial description of the

data content, e.g., items/datasets that do not follow official name nomenclature) the quality of the produced recommendations decreased significantly. Thus DW with the collaborative filtering part turned off only produced an average of about 9% successful recommendations. This shows that the lack of the data description brings with it a noise to the estimated weights for the recommendations, creating an over-fitting problem. This also provides a reasoning to use the content-based filtering approach as an extension to the collaborative filtering approach when using both approaches as a hybrid.

The above described preferential hybrid approach shows substantial improvements to the quality of the recommendations, reducing the total numbers of recommendations at the same time (i.e., reducing clutter). Figure 6.4 shows the corresponding comparison of the provided and followed recommendations; the achieved average ratio of TP to FP has risen to about 30%.

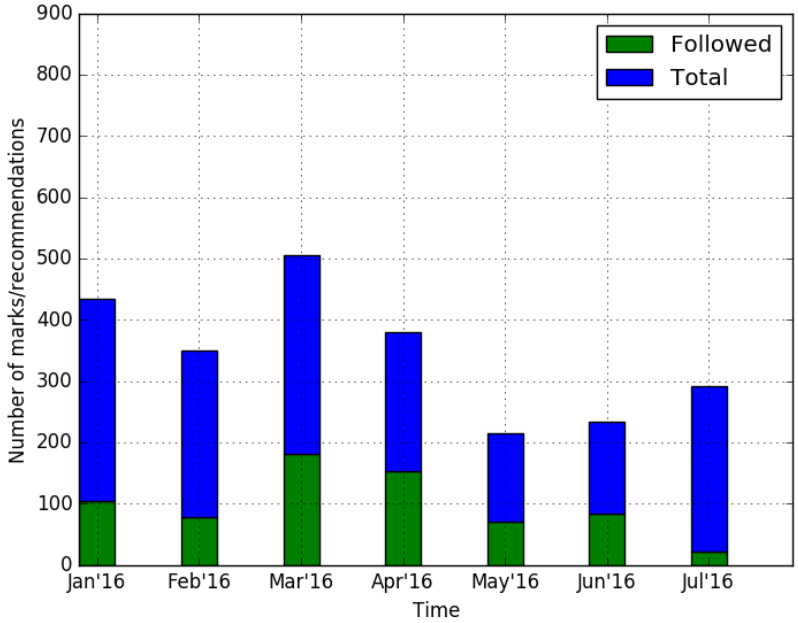


Figure 6.4: Comparison of the provided and followed recommendations per month (using collaborative filtering with content-based approach)

The results obtained show that a predicted hidden relationships between user data needs indeed exist. This relationship can be revealed by the appropriate combination of machine learning techniques as employed by DW. DW then enables the identification of data according to the needs of a group of users, and provides an estimate of the number of future data accesses. We have also shown that is also important to consider the content of data that reveals relationships between user and data attributes (another dimension of user interests). It is important to note that the statistics collected were not based on actual user interaction of the system, thus DW could not make use of implicit and explicit user feedback (as users have not provided such feedback) when generating recommendations. Thus, we argue that these results represent the minimum of what DW could achieve if it was indeed used by the majority of PanDA users.

6.3 User Experience

Data Watcher extends users' abilities in discovering new data, and in being aware of what data is used by other users (especially users who worked on the similar data in the past). During the analysis the following concern related to the rating system was raised: due to the high number of data items in the system, it is difficult to engage users to rate a particular item. Thus users with a small number of used items (e.g., dozens) could provide data ratings per item with much less effort, compared to users with large number of used items (e.g., hundreds or thousands of used items); the latter part of users would likely prefer to rate data on a per collection basis (i.e., group of items).

Data Watcher provides a possibility to produce recommendations manually from one user to another. For example if user A provides a recommendation to user B (i.e., mark data for user B) and user B indeed takes this recommendation into consid-

eration (i.e., apply action "consider"), then the system can increase the weight for the data in the recommendation and can send this information to WMS. Such behavior could trigger so called "lazy data transfers", transfers with lower priority, but with specified destination (that would be determined by the Brokerage subsystem based on the data type), to prepare the system for future user interest in that particular data.

6.4 Summary

In this chapter we explored the quality of recommendations (i.e., the list of items with the highest predicted ratings) that DW is able to provide based on the chosen machine learning filtering approach. The hybrid approach (collaborative filtering with a content-based filtering extension) showed a significant improvement over of the component approaches. We have also reviewed what kind of improvement in the user experience can be expected by the users of DW.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

In this final chapter of the dissertation, we will provide quick review to the conducted research; we will discuss the expected benefits to users and to the applicable WMS (in particularly, PanDA WMS). We will devote the last section to future work; more precisely, we will describe ideas to the possible improvement of the Data Watcher system, and its analysis and analytics processes.

7.1 Conclusions

The original goal of our research was to improve the data need foresight in large scale scientific computing workload management systems by determining which data will most likely be in demand by users and thus enabling the data distribution system to create an appropriate number of replicas. During our preliminary investigations into the topic we have realized that, in general, user as a detached entity is ignored (only could be considered as one of the parameters) even in approaches that try to model data demand. We believed that an explicit investigation of user behavior could significantly advance the foresight to data popularity. Thus the direction we have chosen is user-centric; more precisely, we proposed to build a user model which will represent user preferences and will help to predict future user interests (and behavior), thus to estimate the popularity of data (and its significance) for a defined group of users.

The Production and Distributed Analysis system PanDA is a pilot-based WMS that operates at the ATLAS experiment at CERN, and uses resources of the WLCG

infrastructure. PanDA manages jobs execution on the grid and heavily relies on the ATLAS Distributed Data Management system (Rucio), since among others its scheduling heavily depends on the initial data distribution. Our work is focused on PanDA with an intent to improve PanDA's brokering process (provide estimated data weights that represent user future interests) as well as to enhance user experience.

The first step in our research was to determine whether or not there is a correlation between users and/or user-data usage in historical PanDA usage data. Analysis of archived PanDA usage data via data mining methods indeed showed that existing data usage patterns could be utilized to predict future user activity. Thus a recommender system approach was selected as a suitable technique for the personalization and exploration of user-data relationships using machine learning techniques. As an outcome, the Data Watcher system was designed and implemented. During the analysis of the archival data and during the design of DW it has become more and more apparent that modeling of user behavior would likely have a profound effect on user experience and not just on data distribution. DW was designed with this aim in mind.

DW provides valuable information about relationships between users and their data needs, and aims to enhance user experience by providing a possibility to discover new data based on similarity characteristics between users and between data itself. As a hub for the user-data relationships, DW also provides a platform for users to share their feedback (as a quantitative assessment) about how a particular data (or data collection) fits their needs; and/or to provide "manual" recommendations for users of the same interest area.

The preliminary results showed that the efficiency of the DW core predictive subsystem, i.e., the recommender system, heavily depends on the exact information filtering approaches employed. We have found that collaborative filtering with a

content-based filtering extension (hybrid approach) showed the highest feasibility in predicting user interest in a particular data with certain level of confidence. The quality of predictions produced by the hybrid approach was shown to be significantly higher than any of its component approaches by themselves. We argued that a wide user-base acceptance of DW would enable PanDA to use better performing data replica placements.

7.2 Future Work

Since this work confirmed that a user behavior modeling approach can lead to better data popularity prediction, future work should be focused on performance enhancements of the Data Watcher analysis process.

One of the proposed enhancements is to extend the content-based filtering component by using deeper associations between attributes. Groups of attribute types could be established to be considered together. Each group then could have a certain weight that is proportional to the number of types in the group, and a certain significance measure of that group among the others. Thus if an item contains attributes with types from one of the predefined groups then these attributes would form a collection of attributes. In this case the user profile would be attribute collection oriented rather than just single attribute based.

In addition to the two main information filtering approaches, there is a possibility to involve a rule-based approach that could be built around sequential pattern mining technique (see Chapter 4). Sequential pattern mining techniques could extract sequential patterns from users' activities (sets of data used during certain time periods that are ordered).

Further improvements in Data Watcher could heavily influence the extent to which PanDA job- and data-placement decisions are made, and could result in significant improvement in user experience.

APPENDIX A

DATA WATCHER DATABASE SCHEMA

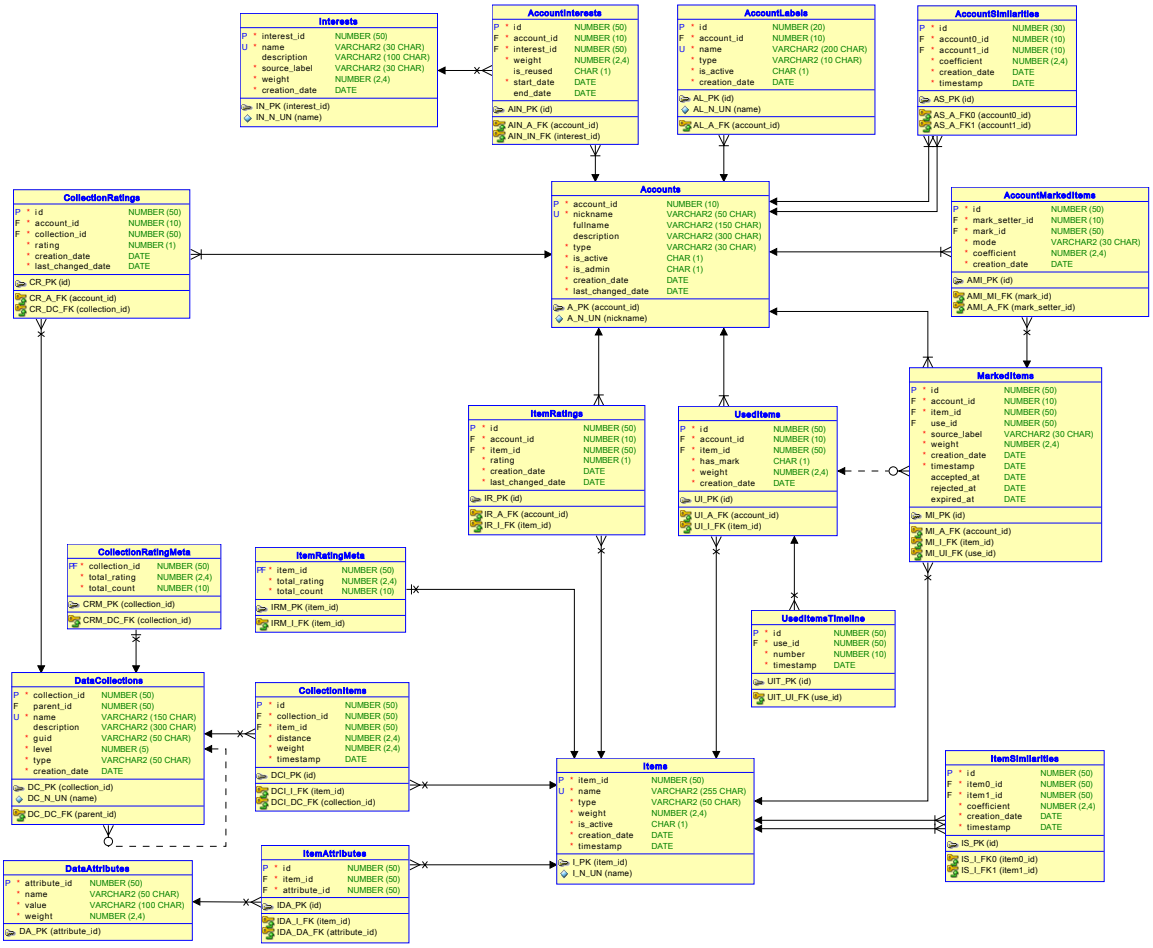


Figure A.1: Data Watcher database schema

REFERENCES

[1] ATLAS Collaboration, "The atlas experiment at the cern large hadron collider," *Journal of Instrumentation*, vol. 3, no. 08, 2008.

[2] I. Bird, P. Buncic, F. Carminati, M. Cattaneo, P. Clarke, I. Fisk, M. Girone, J. Harvey, B. Kersevan, P. Mato, R. Mount, and B. Panzer-Steindel, "Update of the computing models of the wlcg and the lhc experiments," 2014. [Online]. Available: https://cds.cern.ch/record/1695401

[3] G. Duckeck, D. Barberis, R. Hawkings, R. Jones, N. McCubbin, G. Poulard, D. Quarrie, T. Wenaus, and E. Obreshkov, *ATLAS Computing: Technical design report*, ser. Technical Design Report ATLAS. CERN, 2005. [Online]. Available: http://cds.cern.ch/record/837738

[4] S. McKee, "The atlas computing model: status, plans and future possibilities," *Computer Physics Communications*, vol. 177, no. 12, pp. 231–234, 2007.

[5] F. B. Megino, K. De, J. Caballero, J. Hover, A. Klimentov, T. Maeno, P. Nilsson, D. Oleynik, S. Padolski, S. Panitkin, A. Petrosyan, and T. Wenaus, "Panda: Evolution and recent trends in lhc computing," in *Procedia Computer Science*, vol. 66, 2015, pp. 439–447.

[6] C. Serfon, M. Barisits, T. Beermann, V. Garonne, L. Goossens, M. Lassnig, A. Nairz, and R. Vigne, "Rucio, the next-generation data management system in atlas," *Nuclear and Particle Physics Proceedings*, vol. 273-275, pp. 969–975, 2016.

[7] M. Barisits, C. Serfon, V. Garonne, M. Lassnig, T. Beermann, and R. Vigne, "Resource control in atlas distributed data management: Rucio accounting and quotas," *Journal of Physics: Conference Series*, vol. 664, no. 6, 2015.

[8] C. Serfon, M. Barisits, T. Beermann, V. Garonne, L. Goossens, M. Lassnig, A. Molfetas, A. Nairz, G. Stewart, and R. Vigne, "Atlas dq2 to rucio renaming infrastructure," *Journal of Physics: Conference Series*, vol. 513, no. 4, 2014.

[9] K. De, A. Klimentov, S. Panitkin, M. Titov, A. Vaniachine, T. Wenaus, D. Yu, and G. Záruba, "Poster: Panda: Next generation workload management and analysis system for big data," *High Performance Computing, Networking Storage and Analysis, SC Companion*, 2012.

[10] T. Maeno, K. De, T. Wenaus, P. Nilsson, R. Walker, A. Stradling, V. Fine, M. Potekhin, S. Panitkin, and G. Compostella, "Evolution of the atlas panda production and distributed analysis system," *Journal of Physics: Conference Series*, vol. 396, no. 3, 2012.

[11] P. Nilsson, "Experience from a pilot based system for atlas," *Journal of Physics: Conference Series*, vol. 119, no. 6, 2008.

[12] M. Titov, G. Záruba, A. Klimentov, and K. De, "A probabilistic analysis of data popularity in atlas data caching," *Journal of Physics: Conference Series*, vol. 396, no. 3, 2012.

[13] T. Beermann, P. Maettig, G. Stewart, M. Lassnig, V. Garonne, M. Barisits, R. Vigne, C. Serfon, L. Goossens, A. Nairz, and A. Molfetas, "Popularity prediction tool for atlas distributed data management," *Journal of Physics: Conference Series*, vol. 513, no. 4, 2014.

[14] T. Beermann, G. Stewart, and P. Maettig, "A popularity based prediction and data redistribution tool for atlas distributed data management," *PoS*, vol. ISGC2014, 2014.

[15] V. Kuznetsov, T. Li, L. Giommi, D. Bonacorsi, and T. Wildish, "Predicting dataset popularity for the cms experiment," in *17th International workshop on Advanced Computing and Analysis Techniques in physics research (ACAT 2016)*, 2016.

[16] D. Bonacorsi, V. Kuznetsov, T. Wildish, and L. Giommi, "Exploring patterns and correlations in cms computing operations data with big data analytics techniques," *PoS*, vol. ISGC2015, 2015.

[17] F. B. Megino, M. Cinquilli, D. Giordano, E. Karavakis, M. Girone, N. Magini, V. Mancinelli, and D. Spiga, "Implementing data placement strategies for the cms experiment based on a popularity model," *Journal of Physics: Conference Series*, vol. 396, no. 3, 2012.

[18] M. Hushchyn, P. Charpentier, and A. Ustyuzhanin, "Disk storage management for lhcb based on data popularity estimator," *Journal of Physics: Conference Series*, vol. 664, no. 4, 2015.

[19] M. Turilli, M. Santcroos, and S. Jha, "A comprehensive perspective on the pilot-job systems," *CoRR*, vol. abs/1508.04180v3, 2015.

[20] A. Luckow, M. Santcroos, O. Weidner, A. Merzky, P. K. Mantha, and S. Jha, "P*: A model of pilot-abstractions," *CoRR*, vol. abs/1207.6644, 2012.

[21] A. Luckow, M. Santcroos, A. Zebrowski, and S. Jha, "Pilot-data: An abstraction for distributed data," *Journal of Parallel and Distributed Computing*, vol. 7980, pp. 16–30, 2015.

[22] A. Luckow, L. Lacinski, and S. Jha, "Saga bigjob: An extensible and interoperable pilot-job abstraction for distributed applications and systems," in *Cluster, Cloud and Grid Computing (CCGrid)*, 2010, pp. 135–144.

[23] A. Rajaraman and J. D. Ullman, *Mining of Massive Datasets*. Cambridge University Press, 2011.

[24] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, 2nd ed. Morgan Kaufmann Publishers, 2006.

[25] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Pearson, 2005.

[26] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proceedings of the 20th International Conference on Very Large Data Bases*, ser. VLDB '94. Morgan Kaufmann Publishers Inc., 1994.

[27] T. Slimani and A. Lazzez, "Sequential mining: Patterns and algorithms analysis," *International Journal of Computer and Electronics Research*, vol. 2, no. 5, 2013.

[28] M. J. Zaki, "Spade: An efficient algorithm for mining frequent sequences," *Machine Learning*, vol. 42, 2001.

[29] F. Hernández del Olmo and E. Gaudioso, "Evaluation of recommender systems: A new approach," *Expert Systems with Applications*, vol. 35, no. 3, pp. 790–804, 2008.

[30] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, *Recommender Systems Handbook*. Springer, 2011.

[31] J. Leskovec, A. Rajaraman, and J. D. Ullman, *Mining of Massive Datasets*. Cambridge University Press, 2014.

[32] B. Mobasher, "Data mining for web personalization," *The Adaptive Web: Lecture Notes in Computer Science*, vol. 4321, pp. 90–135, 2007.

[33] C. Goutte and E. Gaussier, "A probabilistic interpretation of precision, recall and f-score, with implication for evaluation," *27th European Conference on IR Research, ECIR 2005, Santiago de Compostela*, vol. 3408, pp. 345–359, 2005.

[34] G. Schröder, M. Thiele, and W. Lehner, "Setting goals and choosing metrics for recommender system evaluations," in *CEUR Workshop Proceedings*, vol. 811, 2011, pp. 78–85.

[35] P. Brusilovsky, A. Kobsa, and W. Nejdl, *The Adaptive Web*. Springer-Verlag Berlin Heidelberg, 2007.

[36] M. Hahsler, B. Grün, and K. Hornik, "Introduction to arules: Mining association rules and frequent item sets," *SIGKDD Explorations*, vol. 2, 2007.

[37] M. Hahsler, S. Chelluboina, K. Hornik, and C. Buchta, "The arules r-package ecosystem: Analyzing interesting patterns from large transaction data sets," *Journal of Machine Learning Research*, vol. 12, 2011.

[38] P. Fournier-Viger, A. Gomariz, T. Gueniche, A. Soltani, C.-W. Wu, and V. S. Tseng, "Spmf: A java open-source pattern mining library," *Journal of Machine Learning Research*, vol. 15, 2014.

[39] P. Fournier-Viger, A. Gomariz, M. Campos, and R. Thomas, *Advances in Knowledge Discovery and Data Mining*. Springer International Publishing, 2014, ch. Fast Vertical Mining of Sequential Patterns Using Co-occurrence Information.

[40] M. Titov, "Python package aimed to explore relationships in analyzed data," https://github.com/mtitov/pyrexplorer.

BIOGRAPHICAL STATEMENT

Mikhail Titov was born in Moscow, Russian Federation (former USSR), in 1984. He received his Specialist (B.S. and M.S.) degree from the National Research Nuclear University MEPhI (Moscow Engineering Physics Institute), Russian Federation, in 2007, in Computer Engineering with the specialization "Computers, Computer Systems and Networks", his Ph.D. degree from the University of Texas at Arlington in 2016, in Computer Science. From 2003 to 2012, he was with the department of Information Technologies, Moscow Engineering Physics Institute as a system administrator and engineer-programmer. In 2008, he joined the ATLAS collaboration at CERN (the European Organization for Nuclear Research) as a software developer.