A study of the applicability of recommender systems for the Production and Distributed Analysis system PanDA of the ATLAS Experiment

M Titov\textsuperscript{1}, G Záruba\textsuperscript{2}, K De\textsuperscript{2}, A Klimentov\textsuperscript{3}, S Jha\textsuperscript{4} on behalf of the ATLAS Collaboration

\textsuperscript{1} National Research Centre “Kurchatov Institute”, Moscow, Russia
\textsuperscript{2} University of Texas at Arlington, Arlington, TX, USA
\textsuperscript{3} Brookhaven National Laboratory, Upton, NY, USA
\textsuperscript{4} Rutgers, The State University of New Jersey, New Brunswick, NJ, USA
E-mail: mikhail.titov@cern.ch

Abstract. Scientific computing has advanced in the ways it deals with massive amounts of data, since the production capacities have increased significantly for the last decades. Most large science experiments require vast computing and data storage resources in order to provide results or predictions based on the 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, but also of individual users' interests in the distributed data (reveal implicit interconnection between user and data objects). This however requires the collection and use of specific statistics such as correlations between data distribution, the mechanics of data distribution, and mainly user preferences. This work focuses on user activities (specifically, data usages) and interests in such a distributed computing system, namely PanDA (Production ANd Distributed Analysis system). PanDA is a high-performance workload management system originally designed to meet production and analysis requirements for a data-driven workload at the Large Hadron Collider Computing Grid for the ATLAS Experiment hosted at CERN (the European Organization for Nuclear Research). In this work we are going to investigate whether data collection that was gathered in the past in PanDA shows any trends indicating that users could have mutual interests that would be kept for the next data usages (i.e., data usage patterns), using data mining techniques such as association analysis, sequential pattern mining, and basics of the recommender system approach. We will show that such common interests between users indeed exist and thus could be used to provide recommendations (in terms of the collaborative filtering) to help users with their data selection process.

1. Introduction

For a distributed data analysis system to work, the user-selected data has to be available together with the user’s executables at a compute node; more precisely, this distribution of executables and data is one of the most important tasks of such systems. In this paper we are going to focus on PanDA (Production ANd Distributed Analysis system). PanDA is a high-performance workload management system originally designed to meet production and analysis requirements for a data-driven workload at the Large Hadron Collider Computing Grid for the ATLAS (A Toroidal LHC ApparatuS) experiment \cite{1} hosted at CERN (the European Organization for Nuclear Research) in Switzerland.
For a system like PanDA to work efficiently, data (usually large) with high user demand should be preloaded at several computing clusters so that job executables (usually small) can be scheduled with fewer restrictions and shorter turnaround times. It is thus to the advantage of PanDA to explicitly collect information on how “hot” (with high demand) datasets are in order for it to be able to correlate the number of replicas over the network. One way to predict the need for an individual dataset is to determine whether it is being used by a user who has shared interests (in other datasets) with a number of other users. Thus, the goal of such analysis would be to perform an evaluation of correlations between users’ interests and similarities in data usage between correlated users. The outcome of such analysis could reveal what impact a recommender system would have if it was integrated into PanDA. Currently in ATLAS, sharing interest in datasets is usually done by word of mouth or over emails and blogs.

An earlier study [2] of data popularity with stochastic methods based on information from PanDA (job parameters of processed data from the ATLAS experiment) required a highly complex model design; yet it only showed insignificant 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.

1.1. PanDA system
Production and Distributed Analysis system PanDA is a pilot-based workload management system. This means, that workload is assigned based on the feedback from successfully activated and validated pilot jobs, which are lightweight processes that probe the environment (on compute nodes and clusters) and act as “smart wrappers” for the payload. (Generally, pilots manage data and executables on compute nodes they are running on.) In PanDA, an independent subsystem manages the delivery of pilot jobs to all worker nodes via a number of cluster and grid scheduling systems (e.g., Condor-G). Pilot based systems like PanDA also enable the integration of non-grid based resources and thus can scale and work simultaneously on large clusters as well as individual computers (by using local pilot submission factories).

Jobs are defined by a set of specifications, their associated datasets and the input/output files within them. The PanDA server places such jobs into a global job queue, upon which a brokerage module operates to prioritize and assign work on the basis of job type, priority, input data and its locality, available CPU resources, and other brokerage criteria. Allocation of job sets to computer centers is followed by the dispatch of corresponding input datasets to those centers or nearby ones, handled by a data service system located within the ATLAS distributed data management system.

2. Data Analysis Techniques

2.1. Data Mining
Data mining, in general, is considered as the analysis step of knowledge discovery in databases (KDD), and is defined as the application of specific algorithms for extracting patterns from data [3]. Its functionality is used to specify the kind of patterns to be found in data mining tasks [4]. Such data mining tasks can be classified into two categories: descriptive and predictive. Descriptive mining tasks characterize the general properties of the data in the database. Predictive mining tasks perform inference on the current data in order to make predictions [4].

In this paper, we focus on descriptive mining tasks, and thus the goal of our mining process is to capture, model, and analyze the behavioral patterns and profiles of users interacting with data (i.e., the discovery of associations and correlations within PanDA data).

2.2. Data Modeling and Representation
Transactional form is used to represent data for analysis. Set of transactions are as following $T = \{t_1, t_2, \ldots, t_M\}$, 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 $t_m$. Since a transaction is supposed to describe which items are requested during a certain time window by a user, we restrict the duration of transactions to a common transaction time window $\Delta t$. (At the creation of the transaction database, if an activity is longer than $\Delta t$ then it gets split into several $\Delta t$ 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 used to define a user-specific function that calculates the degree of interest of a user in a specified item and can be seen as the user profile. The set of user profiles is thus $\mathbb{UP} = \{up_1, up_2, \ldots, up_K\}$ and $interest\_degree(k,n) = up_k(i_n)$.

2.3. 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. Thus our objective is to identify item pairs in that co-occur more frequently than an analyst-predicted value in the transactional database.

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 $X \cap Y = \emptyset$. Association rules are rules that surpass analyst-specified minimum support and minimum confidence thresholds. The support $supp(X)$ of 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 that contain the defined set of items: $supp(X) = |\{t_m \mid X \subseteq I(t_m), t_m \in \mathbb{T}\}| \times |\mathbb{T}|^{-1}$; the confidence of rule $X \Rightarrow Y$ determines how frequently items in $Y$ appear in transactions that contain $X$: $conf(X \Rightarrow Y) = supp(X \cup Y) \times supp(X)^{-1}$. Therefore $X \Rightarrow Y$ will be used as an association rule if it satisfies analyst-predefined minimum support ($\sigma$) and minimum confidence limits ($\delta$):

$$supp(X \cup Y) \geq \sigma \quad \text{and} \quad conf(X \Rightarrow Y) \geq \delta$$

(1)

2.4. Association Rule Mining Algorithms

A common strategy for association rule mining algorithms is to decompose the problem into two major subtasks [5]: i) 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); ii) 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 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 algorithm [8]. 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.
2.5. 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 and/or the longest (with the most number of items and/or with the most number of transactions) sequential patterns with 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 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.

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. Algorithm SPADE [7] was identified to best suit our needs as it lends itself to easy modifications (to satisfy our custom criteria) and 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).

3. Problem Statement
Given the per user data usage of 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 then 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 the 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 users’ future activities (data usage) will likely lead to a usable recommender system for users.

4. Data Study
The previous section does not only provide a statement of the problem but also outlines a path leading to a solution. Distributed computing systems provide information about users, their jobs, and data that is used as the input during the processing of these jobs. As mentioned earlier, PanDA collects such information as records of jobs with references to user (i.e., job’s owner, prodUserId) and data patterns (i.e., job’s input data, prodDBlock). In PanDA, the historical data for the year 2011 (when there was no decision making subsystem of PanDA in production) contains about 220M records. A preprocessing application we have written extracted 1,814 users (1,597 relevant users) and 380,111 items (220,867 relevant items). (Relevant users are defined as users who had analysis jobs that terminated successfully; relevant items have a similar definition related to success and parsability.)
The corresponding statistical data showed that 95.6% (1,526 users) of all relevant users had requested at least two distinct items; that 68.9% of relevant users (1,100 users) had requested more than 10% of the average number of requested items; and that 47.6% of relevant items (105,097 items) were requested by at least two distinct users. In this paper we ignore items that have only been used by one user; however, in a future 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 numbers 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. 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. 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 similarity between users, i.e., similarity in data usage.

4.1.1. Problem Definition  Every user \( u_k \in \mathbb{U} \) has three associated subsets:

- Used items \( UI(u_k) = \{ i_j : [(tu_{kj}, n_{kj}), \ldots] \mid i_j \in I \} \), 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 a sequence of \( (tu_{kj}, n_{kj}) \) pairs associated with it.)

- Similar users \( SU(u_k) = \{ u_m : [(ts_{km}, sc_{km}), \ldots] \mid u_m \in \mathbb{U} \} \), where \( ts_{km} \) is timestamp, time when similarity between two users was evaluated by the analysis system; \( sc_{km} \) is a similarity coefficient between users \( u_k \) and \( u_m \). (When \( sc_{km} \) is updated then \( ts_{km} \) is also updated.)

- Recommended items \( RI(u_k) = \{ i_j : [u_m : [(tr_{jm}, sc_{km}), \ldots] \mid u_m \in SU(u_k) \} \mid i_j \in I \} \), where \( tr_{jm} \) is a timestamp, i.e., time when user \( u_m \) had used item \( i_j \); and \( sc_{km} \) is a similarity coefficient between users \( u_k \) and \( u_m \) at the moment of \( tr_{jm} \). Thus this set contains items that user \( u_k \) may receive as recommended based on similar interests.

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.

4.1.2. 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: i) Maintain \( UI(u_k) \): list of used items is 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); ii) 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 2) as the similarity coefficient, if this coefficient exceeds a threshold value; iii) Maintain \( RI(u_k) \): create new recommendations by inserting all used items of just established similar users into \( RI(u_k) \).
\[ jsc(u_k, u_m) = \frac{|UI(u_k) \cap UI(u_m)|}{|UI(u_k) \cup UI(u_m)|} = \frac{\text{num common used items}}{\text{num total used items}} \] (2)

Figure 1 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 on which our simulation analysis is based was not influenced by recommendation-induced item usages. As we are interested in establishing possible relationships between 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.

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

4.2. 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 \( \text{\texttt{R}} \) (a computational R library for mining association rules and frequent itemsets) by using the Apriori algorithm. Table 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.

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.

4.3. Frequent Sequences in Data

The association analysis in Section 4.2 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 for the provided PanDA data).

There exist 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 \( \text{\texttt{R}} \); and ii) SPMF (Sequential Pattern Mining Framework) - an open-source data mining library written in Java, specialized in pattern mining \( \text{\texttt{R}} \). Both tools showed good results with certain portions of our data, but not with the entire data set. The number of sequences in conjunction with their sheer sizes were
Table 1. Numbers for association rules (generated with R library “arules”).

| Transaction time window (TTW), days | Number of transactions established | Minimum support | Minimum confidence | Number of associated items found | Number of associated rules created |
|-------------------------------------|-----------------------------------|-----------------|--------------------|----------------------------------|-----------------------------------|
| 30                                  | 7,299                             | 0.035           | 0.5                | 71                               | 43,446                            |
|                                     |                                   | 0.0335          | 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                        |

causing frequent crashes of 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 SPADE [7] (Apriori-based). In addition to an implementation of SPADE, we extended it with CMAP (Co-occurrence MAP) [11] and bit sequence representation; this tailoring of SPADE was done to increase the computational efficiency. Our implementation in Python is available at Ref. [12].

![Figure 2](image)

(a) Length of sequences per user  (b) Number of transactions per user

Figure 2. Maximum data usage overlap per user

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 reveals 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 2 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.

5. Conclusion
In this work we investigated whether data that was gathered in the past in PanDA shows any trends that indicate that users could have mutual interests.

Initial analysis of data usages confirmed that a certain percentage of recommendations from similar users are actually followed despite the low quality of those recommendations. Thus user activity can follow some usage pattern of the group of similar users and be correlated with specific user interests. Association analysis showed that correlation between items existed; however, even to spot small signs of these correlations took significant computational cost. To be able to analyze PanDA data further, 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. Our current efforts are focused on an actual implementation of a recommender system that will be closely integrated with the PanDA system thus enabling ATLAS users to benefit early and frequently from these identifications of similarities.

Acknowledgments
We would like to thank the PanDA team for providing the data for our analysis and for their continued support, and to express our appreciations to CERN, UTA and NRC KI for the provided facilities. In Addition, this work has been carried out using computing resources of the federal collective usage center Complex for Simulation and Data Processing for Mega-science Facilities at NRC “Kurchatov Institute”, http://ckp.nrcki.ru/. This work was funded in part by the Russian Ministry of Science and Education under contract No. 14.Z50.31.0024.

References
[1] ATLAS Collaboration 2008 JINST 3
[2] Titov M, Záruba G, Klimentov A and De K 2012 Journal of Physics: Conference Series 396
[3] Rajaraman A and Ullman J D 2011 Mining of Massive Datasets (Cambridge University Press)
[4] Han J and Kamber M 2006 Data Mining: Concepts and Techniques 2nd ed (Morgan Kaufmann Publishers)
[5] Tan P N, Steinbach M and Kumar V 2005 Introduction to Data Mining (Pearson)
[6] Agrawal R and Sikrant R 1994 Proceedings of the 20th International Conference on Very Large Data Bases VLDB '94 (Morgan Kaufmann Publishers Inc.)
[7] Zaki M J 2001 Machine Learning 42
[8] Hahsler M, Grün B and Hornik K 2007 SIGKDD Explorations 2
[9] Hahsler M, Chelluboina S, Hornik K and Buchta C 2011 Journal of Machine Learning Research 12
[10] Fournier-Viger P, Gomariz A, Gueniche T, Soltani A, Wu C W and Tseng V S 2014 Journal of Machine Learning Research 15
[11] Fournier-Viger P, Gomariz A, Campos M and Thomas R 2014 Advances in Knowledge Discovery and Data Mining (Springer International Publishing) chap Fast Vertical Mining of Sequential Patterns Using Co-occurrence Information
[12] Titov M Data relations discovery package https://github.com/mtitov/pyrexplorer