A Running Pattern Recognition Algorithm for Publish/Subscribe Distributed Systems

Wenjun Wu, Zhuowei Shen* and Defeng Wang
School of Cyber Science and Engineering, Southeast University, Nanjing, China

Abstract. Publish/subscribe distributed systems are often used in critical applications. It is necessary to monitor their running patterns in real time to detect abnormal status. Therefore, identifying the normal running pattern is the precondition of monitoring publish/subscribe distributed systems. Based on Apriori algorithm, this paper presents a weighted frequent itemset mining algorithm for running pattern recognition of publish/subscribe distributed systems. By introducing the transaction matrix, the algorithm only needs to scan the transaction database once. By weighting the items from two aspects of influence and frequency, the support of the items with few occurrences but much importance can be improved, so that the running pattern containing small frequency events can be mined out. Experimental results show that the algorithm can effectively mine the running patterns, and has better performance than Apriori algorithm and FP-growth algorithm.

1 Introduction

Publish/subscribe distributed systems based on data distribution service (DDS) [1] are widely used in critical application areas, such as industrial automation, autopilot, military industry. It is necessary to monitor the systems in real time to detect the abnormal status in time, and ensure the application safety. There are usually several running patterns in these systems. The general strategy of real-time monitoring is to identify which pattern the system is running in according to the current running state. If it does not belong to any known normal pattern, it is judged that the system is in an abnormal status. Therefore, it is the precondition and foundation of abnormal state monitoring to identify the normal running patterns. Pub/sub distributed systems are usually large and the interactions between their components are often complex. A monitoring tool for these systems is independent of these systems and does not understand their business logics. So it is particularly difficult to identify the running patterns in these systems.

Apriori algorithm [2,3] is a classic algorithm to mine frequent itemsets. FP-growth algorithm improves the efficiency of Apriori algorithm by reducing the number of database scan [4]. These two algorithms are based on the support threshold mining frequent itemsets, and on the premise that the importance of each item in the database is the same and evenly distributed [5]. However, in most systems, each pattern has different running time and importance. With usual support mining, it is possible that the running pattern with short running time but very important cannot be mined. In this paper, an improved Apriori algorithm, which is named as Weighted Frequent Itemsets Mining (WFIM) algorithm is
proposed. With the pub/sub event log collected by pub/sub middleware, the WFIM algorithm can get the normal running patterns of the pub/sub system, which is of great significance to the anomaly detection of the system.

2 Process of running pattern mining

The monitoring of pub/sub distributed systems is divided into modeling phase and detection phase. In the modeling phase, the knowledge base is constructed by mining running patterns, which can be used for pattern comparison in detection phase.

The running pattern mining mainly includes the following three parts: Data acquisition, preprocessing, and pattern mining.

In data acquisition part, the log of communication events between components is collected by pub/sub middleware.

Then in preprocessing part, the local logs of all the components are collected together to build the publish/subscribe relationships between components. According to the same topic and message, and corresponding publish and subscribe actions, publish/subscribe relationships between components can be find out and form publish/subscribe events, which can be represented as the following 5-tuple: $(\text{Timestamp of publish}, \text{Timestamp of subscribe}, \text{topic}, \text{publisher}, \text{subscriber})$, where publisher and subscriber are marked as the corresponding component IDs. The event set consists of all the pub/sub events.

From the event set, each publish/subscribe relationship can be identified as a triple $(\text{Topic}, \text{publisher}, \text{subscriber})$ and tagged as $L_i$. All publish/subscribe relationships constitute the event tag table $I = \{I_1, I_2, ..., I_m\}$.

According to the topics in the event tag table and its corresponding publishers and subscribers, we can draw the topological graph of the publish/subscribe relationships of the system, and get the event sequence from the event set by marking every event with corresponding tag and sorting them in chronological order.

In pattern mining part, the event sequence is segmented. Each subsequence is treated as a transaction to build a transaction database. With the transaction database, the weighted frequent itemset mining algorithm is used to mine patterns.

3 The weighted frequent itemset mining algorithm

Apriori algorithm is inefficient due to traversing the transaction database more than once. In addition, on the premise of the same importance of items, it completely depends on the support threshold. If the threshold is set too high, the itemsets with lower frequency will be lost. However, in most systems, items correspond to events and item sets correspond to running patterns. The influence of different events is different, and the running time of each running pattern is different, and the frequency of events is also different. As a result, mining according to support will lead to the loss of short running time but important running patterns.

Therefore, this paper proposes a WFIM algorithm, introduces the concept of item weight, reflects the importance of items from two aspects of frequency and influence, further obtains the weight of transactions, uses weighted support mining, and combines with transaction matrix to improve the mining efficiency.

3.1 Definitions

Let $I = \{I_1, I_2, ..., I_m\}$ is a collection of items, corresponding to the collection of publish/subscribe event tags in the system. Each transaction $T$ in transaction database $D$, ...
which is a subsequence of the event sequence, is a subset of itemset \( I \), \( T \subseteq I \).

**Definition 1.** Transaction Matrix \( M \)

\[
M = \begin{bmatrix}
I_{11} & I_{12} & \cdots & I_{1m} & C(T_1) \\
I_{21} & I_{22} & \cdots & I_{2m} & C(T_2) \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
I_{n1} & I_{n1} & \cdots & I_{nm} & C(T_n)
\end{bmatrix}
\]

(1)

Where \( I_{ij} = \begin{cases} 
1, & I_j \in T_i \\
0, & I_j \notin T_i
\end{cases} \), \( i = 1, 2, \ldots, n \), \( j = 1, 2, \ldots, m \). \( C(T_i) \) represents the times transaction \( T_i \) appears in the transaction database \( D \).

By scanning the original transaction database, transactions are de-duplicated and counted, then are stored in matrix \( M \). If there are many duplicate transactions in the transaction database, the original transaction database can be greatly compressed\(^{[6,7]}\).

When counting the occurrence frequency of item \( I_j \), i.e. \( \text{Count}(I_j) \), we only need to calculate \( \text{Count}(I_j) = \sum_{k=1}^{n} I_{kj} \times C(T_k) \). When judging whether a transaction contains an itemset, we only need to do an "AND" operation on the values of the corresponding items in this row. If the result is 1, the transaction contains the itemset, if it is 0, it does not. For example, if we need to determine whether the transaction \( T_i \) contains the itemsets \( t = \{I_{1}, I_{2}\} \), then calculate the value of \( I_{i1} \wedge I_{i2} \).

**Definition 2.** Weight of the Item

The weight of an item is related to the frequency of the item appearing in the transaction database and the influence degree of the item itself. The weight of the item \( I_j \) is defined as:

\[
w(I_j) = \alpha \times \text{E}(I_j) + \beta \times \frac{1}{\text{P}(I_j)}
\]

(2)

Where \( \text{E}(I_j) \) is the influence degree of item \( I_j \), \( \text{P}(I_j) \) represents the proportion of item \( I_j \) that appears in the transaction database. \( \alpha \) and \( \beta \) represent the weights of influence degree and occurrence frequency of the item respectively, \( \alpha + \beta = 1 \). The value of \( \alpha \) and \( \beta \) can be adjusted according to the emphasis on influence or occurrence frequency in the actual scene.

The PageRank algorithm is applied to the evaluation of the influence degree \(^{[8]}\). Based on the topology diagram of publish/subscribe relationships of components, the PageRank algorithm is modified to evaluate the influence degree of each component: each node is no longer a web page but a component, and the directed edge changes from a link to a publish/subscribe relationship. The influence degree of a component \( p \), \( \text{CR}(p) \) is calculated as follows:

\[
\text{CR}(p) = \frac{(1-d)}{n} + d \sum_{i=1}^{n} \frac{\text{CR}(p_i)}{C(p_i)}
\]

(3)

Where \( n \) is the total number of components in the system, and \( p_i \) is the \( i \)-th component which subscribe the component \( p \), \( C(p_i) \) denotes the number of the components that subscribe the component \( p_i \). \( d \) is the damping factor.

If the publisher component of corresponding publish/subscribe event of item \( I_j \) is component \( p \), \( \text{E}(I_j) \), the influence degree of item \( I_j \) is defined as \( \text{CR}(p) \).

In equation (2), \( \text{P}(I_j) \) denotes the proportion of item \( I_j \) that appears in the transaction database. It is defined as:

\[
\text{P}(I_j) = \frac{\text{Count}(I_j)}{|D|}
\]

(4)
Where Count(I\textunderscore j) is the occurrence frequency of the item I\textunderscore j in the transaction database, and |D| represents the total number of transactions in the transaction database. In equation (2), the importance of item’s occurrence frequency is reflected with P(I\textunderscore j)’s reciprocal.

**Definition 3. Weight of the Transaction**

The weight of a transaction is the average weight of the items contained in the transaction multiplied by the transaction count. If |T\textunderscore k| represents the number of items in the transaction T\textunderscore k, then the weight of transaction T\textunderscore k is as follows:

\[ W(T_k) = \frac{\sum I_{kj} \cdot w(I_j)}{|T_k|} \times C(T_k) \] (5)

**Definition 4. Weighted Support of the Itemset**

The weighted support of an itemset t is the ratio of the sum of the transactions weights which contains t to the sum of all transactions weights.

\[ S(t) = \frac{\sum_{k=1}^{n} W(T_k)}{\sum_{k=1}^{n} W(T_k)} \] (6)

### 3.2 Algorithm description

The WFIM algorithm can be described as follows:

```plaintext
func WFIM(D, Sup\textsubscript{min}, L) {
    Input: Transaction Dataset D, minimum support Sup\textsubscript{min};
    Output: Frequent Itemsets L with support ≥ Sup\textsubscript{min}
    Scan D, and build Transaction Matrix M;
    for each I\textunderscore j ∈ I do
        Calculate W(I\textunderscore j) according to equation (2);
    for each T\textunderscore k ∈ D do
        Calculate W(T\textunderscore k) according to equation (5);
    Let L = ∅; k = 1;
    Get candidate k-itemsets CL\textsubscript{k} from Transaction Matrix M;
    for each t ∈ CL\textsubscript{k} do  //get frequent k-itemsets L\textsubscript{k}
        Calculate S(t) according to equation (6);
        if S(t) ≥ Sup\textsubscript{min} then L\textsubscript{k} = {t} ∪ L\textsubscript{k}
    while L\textsubscript{k} ≠ ∅ do
        L = L ∪ L\textsubscript{k}; k = k + 1;
        get CL\textsubscript{k} from L\textsubscript{k-1} by connecting and trimming;
        for each t ∈ CL\textsubscript{k} do  //get frequent k-itemsets L\textsubscript{k}
            Calculate S(t) according to equation (6);
            if S(t) ≥ Sup\textsubscript{min} then L\textsubscript{k} = {t} ∪ L\textsubscript{k}
    return L;
}
```

### 4 Experiments

In an automatic pilot control system based on DDS, the application components include several sensors, data processing controllers and actuators to control the driving of vehicles. There are three running patterns: normal driving, braking and turning. The topology of the publish/subscribe relationships of the components are shown in Figure 2. I\textsubscript{1}–I\textsubscript{6} means publish/subscribe events. In normal driving pattern, events \{I\textsubscript{1}, I\textsubscript{2}, I\textsubscript{3}, I\textsubscript{4}\} occur. In braking pattern, events \{I\textsubscript{1}, I\textsubscript{2}, I\textsubscript{3}, I\textsubscript{5}\} occur. And in turning pattern, events \{I\textsubscript{1}, I\textsubscript{2}, I\textsubscript{3}, I\textsubscript{6}\} occur.
In most of the time, the system runs in normal driving pattern, and the proportion of turning and braking time is small. While the system is running, its publish/subscribe log data is collected. After data preprocessing, WFIM algorithm is executed, with $\text{Sup}_{\text{min}} = 0.2$, $\alpha = 0.5$ and $\beta = 0.5$. With the same log data, Apriori algorithm is run to mining the frequent itemsets.

![Figure 1. Topology of publish/subscription relationships.](image)

<table>
<thead>
<tr>
<th>Itemset</th>
<th>WFIM algorithm</th>
<th>Apriori algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_1, I_2, I_3, I_4$ (Driving)</td>
<td>0.529</td>
<td>0.70</td>
</tr>
<tr>
<td>$I_1, I_2, I_3, I_5$ (Braking)</td>
<td>0.262</td>
<td>0.20</td>
</tr>
<tr>
<td>$I_1, I_2, I_3, I_6$ (Turning)</td>
<td>0.209</td>
<td>0.10</td>
</tr>
</tbody>
</table>

The comparison result of 4-itemsets support is shown in Table 1. WFIM algorithm can successfully mine the three running patterns in the system, while the turning pattern cannot be mined out in Apriori algorithm due to its short running time. Moreover, compared with Apriori algorithm, WFIM algorithm improves the support of itemsets containing $I_5$ and $I_6$, and reduces the support of itemsets containing $I_4$. By weighting the items from two aspects of influence and frequency, the support of the items with few occurrences but much importance can be improved, and the running pattern containing small frequency events can be mined out.

In order to verify the performance of WFIM algorithm, the execution time of WFIM, Apriori and FP-growth algorithms are compared. Figure 2 shows the execution time of the three algorithms with different number of transactions when $\text{Sup}_{\text{min}} = 0.2$. Figure 3 shows the execution time of the three algorithms with the same number of transactions and different support.

The results show that the performance of AFIM algorithm is significantly improved compared with Apriori and FP-growth algorithms, and with the increase of transaction number, the gap of running time becomes larger. Moreover, the running time of AFIM algorithm is better than the other two algorithms under each support threshold. This is because AFIM algorithm only scans the transaction database once and the transaction matrix is used to compress the transaction database, which reduces the scanning time, and the AND operation in the matrix also reduces the calculation time of itemset support.
5 Conclusion

In this paper, by weighting the items from two aspects of influence and frequency, a WFIM algorithm is proposed to recognize the running patterns in publish/subscribe distributed systems. The experimental results show that the proposed algorithm can effectively mine the running patterns in the system, and its efficiency is better than Apriori and FP-growth algorithms. In the follow-up work, we will study the parallelization scheme of WFIM algorithm to deal with the massive running data of publish / subscribe distributed system, and further improve the performance of the algorithm.

6 References