An Efficient Parallel Anomaly Detection Algorithm Based on Hierarchical Clustering

Ren Wei-wu, Hu Liang, Zhao Kuo,Chu Jianfeng
College of Computer Science and Technology, Jilin University, Changchun, China
E-mail: Chujf@jlu.edu.cn

***Abstract*—For the purpose of improving real time and profiles accuracy, a parallel anomaly detection algorithm based on hierarchical clustering has been proposed. Training and predicting are two busiest processes and they are parallel designed and implemented. Moreover, an abnormal cluster feature tree is built to dig anomalies from normal profiles. A series of experiment results on well-known KDD Cup 1999 data sets indicate that the improved algorithm has superior performance in both detection and real time.**

***Index Terms*—parallel algorithm; hierarchy clustering; abnormal cluster feature tree; normal profiles**

I. Introduction

With the number of intrusion and hacking incidents around the world on the rise, the importance of having dependable intrusion detection systems in place is greater than ever. An intrusion detection system is designed to detect several types of abnormal behaviors that can compromise the security and trust of a computer system.

Now the main intrusion detection technology is divided into two categories: misuse detection [1-2] and anomaly detection [3-7]. Misuse detection encodes the known attacks into the signatures and detects attacks whose signatures are known and have been encoded. Intrusion detection system based on misuse detection can not detect unknown attacks. The process of signing attacks is enormous cost for systems. Unlike misuse detection, anomaly detection builds the normality profiles on the basis of normal behaviors of users, often using machine learning or data mining techniques. In the process of detection, online traffic is matched with the normality profiles, and deviations are marked as anomalies. Since no knowledge of attacks is used to train the normality profiles, anomaly detection can detect previously unknown attacks. Therefore anomaly detection is hotspot in the field of intrusion detection.

Many popular technologies are applied in the field of anomaly detection. Clustering algorithm [21-23] is a successful application in the field of anomaly detection. Density clustering [9-12] and hierarchical clustering [12-15] are two outstanding representative kinds of clustering algorithms. Density clustering can build arbitrary shape cluster, but calculation is too complex. Hierarchical clustering has good efficiency and is easy to implement incremental algorithm, but only build spherical cluster. So profiles of hierarchical are generally less precise than profiles of density clustering.

Moreover, the growth of network flow makes the real time of anomaly detection algorithms be involved. The old serial algorithm can not meet the real time of detection, and the development of CPU has officially entered the era of multi-core. Along with the update multi-core technology, parallel algorithm must be a new way to solve the problem of real time.

Parallel algorithms [16-20] are valuable because of substantial improvements in multiprocessing systems and the rise of multi-core processors. There are two ways parallel processors communicate, shared memory or message passing. Shared memory processing needs additional locking for the data, imposes the overhead of additional processor and bus cycles, and also serializes some portion of the algorithm. Message passing processing use channels and message boxes but this communication adds transfer overhead on the bus, additional memory need for queues and message boxes and latency in the messages. Designs of parallel processors use special buses like crossbar so that the communication overhead will be small but it is the parallel algorithm that decides the volume of the traffic.

For the purpose of improving real time and profiles accuracy of hierarchical clustering, a parallel anomaly detection algorithm based on hierarchical clustering has been proposed in this paper. Training process and predicting process, which consume a lot of resources, are parallel designed and implemented. Moreover, an abnormal cluster feature tree is built to dig anomalies from normal profiles. It can compensate the lack of profiles accuracy of hierarchical clustering.

The rest of paper is organized as follows. In section 2, we introduce some basic concepts. Section 3 presents details of parallel algorithms. Section 4 presents our experiments results and analysis. Finally, we summarize our conclusions and future work in section 5.

II. Basic Concepts

The related basic concepts are introduced at first:

Definition 0: we assume that each sample point lies in the k-dimensional Euclidean space, and a cluster

Corresponding author: Chu Jianfeng
Email: Chujf@jlu.edu.cn

doi:10.4304/jnw.8.3.672-679

$C = \{\vec{v}_i \mid i = 1 \ldots n\}$ is defined the collection of $\vec{v}_i$, $\vec{v}_i$ is a vector which starts with the origin, and ends with the d_i.

Definition 1: For a cluster $C = \{\vec{v}_i \mid i = 1 \ldots n\}$ and the k-dimensional vector $\vec{v}_i$ in the cluster, its center is defined as:

$$\vec{v}_0 = \frac{\sum_{i=1}^{n} \vec{v}_i}{n} \tag{1}$$

The center describes the distribution of points in the cluster.

Definition 2: for a cluster $C = \{\vec{v}_i \mid i = 1 \ldots n\}$ and the k-dimensional vector $\vec{v}_i$ in the cluster, its radius R(C) is defined as:

$$R(C) = \sqrt{\frac{\sum_{i=1}^{n} (\vec{v}_i - \vec{v}_0)^2}{n}} \tag{2}$$

Radius R(C) describes the clustering degree of points in cluster (i.e., the mean distance from all the points to center $\vec{v}_0$)

Definition 3: for two clusters C1 and C2, and their centers $\vec{v}_0$ and $\vec{v}_0'$, the distance from C1 to C2 is defined as the Euclidean distance from $\vec{v}_0$ to $\vec{v}_0'$.

Definition 4: Cluster Feature (CF) is a triple: $CF = \{n, \vec{s}, ss\}$, among which n is the number of vector $\vec{v}_i$ in the cluster C, and $\vec{s}$ is the linear sum of all the vectors in the cluster C (i.e., $\vec{s} = \sum_{i=1}^{n} \vec{v}_i$), and ss is the sum of squares of all the vectors in the cluster C (i.e., $ss = \sum_{i=1}^{n} \vec{v}_i^{\,2}$). CF describes the overall feature of cluster.

Definition 5: for two clusters C1 and C2, and their sum C1+C2 represents the merger of two clusters (i.e., all the vectors in two clusters are merged into one new cluster)

Theorem 1: for two clusters C1 and C2, and their CF $CF_1 = \{n_1, \vec{s}_1, ss_1\}$ and $CF_2 = \{n_2, \vec{s}_2, ss_2\}$, then CF of C1+C2 is $CF_{1+2} = \{n_1 + n_2, \vec{s}_1 + \vec{s}_2, ss_1 + ss_2\}$.

Proof: the proof of theorem 1 is simple. Due to definition 4 and definition 5, theorem 1 is not difficult to show.

Theorem 2: cluster C and its $CF = \{n, \vec{s}, ss\}$, then its center can be represent as $\vec{v}_0 = \frac{\vec{s}}{n}$

Proof: due to definition 1 and definition 4, $\vec{s} = \sum_{i=1}^{n} \vec{v}_i^{\,2}$, then $\vec{v}_0 = \frac{\vec{s}}{n}$.

Theorem 3: cluster C and its $CF = \{n, \vec{s}, ss\}$, then its radius can be represent as

$$R(C) = \sqrt{\frac{ss}{n} - \frac{\vec{s}^{\,2}}{n^2}} \tag{3}$$

Proof: due to definition 2, $R(C) = \sqrt{\frac{\sum_{i=1}^{n} (\vec{v}_i - \vec{v}_0)^2}{n}}$,

After vector expansion: $R(C) = \sqrt{\frac{\sum_{i=1}^{n} \sum_{j=1}^{d} \left(v_{ij} - v_{0j}\right)^2}{n}}$

$$R(C) = \sqrt{\frac{\sum_{i=1}^{n} \sum_{j=1}^{d} \left(v_{ij}^{\,2} + v_{0j}^{\,2} - 2 v_{ij} v_{0j}\right)}{n}}$$

Then

$$R(C) = \sqrt{\frac{\sum_{i=1}^{n} \sum_{j=1}^{d} v_{ij}^{\,2} + n \sum_{j=1}^{d} v_{0j}^{\,2} - 2\left[\sum_{j=1}^{d} \left(v_{0j} \sum_{i=1}^{n} v_{ij}\right)\right]}{n}}$$

,

Due to $\vec{v}_0^{\,2} = \sum_{j=1}^{d} v_{0j}^2$,

$$R(C) = \sqrt{\frac{\sum_{i=1}^{n} \sum_{j=1}^{d} v_{ij}^{\,2} + n\vec{v}_0^2 - 2\left[n \sum_{j=1}^{d} \left(v_{0j}^2\right)\right]}{n}}$$

Then

$$R(C) = \sqrt{\frac{\sum_{i=1}^{n} \sum_{j=1}^{d} v_{ij}^{\,2} + n\vec{v}_0^2 - 2n\vec{v}_0^2}{n}} = \sqrt{\frac{\sum_{i=1}^{n} \sum_{j=1}^{d} v_{ij}^{\,2} - n\vec{v}_0^2}{n}}$$

Due to definition 4: $\vec{s} = \sum_{i=1}^{n} \vec{v}_i = n\vec{v}_0$ and $\vec{s} = \sum_{i=1}^{n} \vec{v}_i = n\vec{v}_0$, and they are substituted in the above equation, then $R(C) = \sqrt{\frac{ss}{n} - \frac{\vec{s}^{\,2}}{n^2}}$.

It can be seen by the above theorem. The center and radius of cluster can be calculated by its CF without having to know each specific vector. Therefore, the algorithm to run needs not to keep original data.

Cluster Feature tree (CF tree): CF is organized in a tree structure. A CF tree represents the present user's profile. CF tree is height balanced tree. For each leaf node, its height is the same. There are three parameters: the maximum number of branches B, threshold T and the maximum number of cluster M. The node of CF tree is table which contains B entries at most. The entry can be represented by the form ｛CF_i, pChild｝, i = 1, 2, ⋯,B, pChild points to the child node of present node. And CF_i represent the merger of clusters of the child node to which is pointed by pChild. In particular, the leaf node of CF tree is not same with the other nodes. The value of its entry pChild is NULL, and the CF_i does not represent the merger of clusters but specific clusters. Each cluster in the leaf node must satisfy the threshold T (i.e., the radius of each cluster must be less than T). The number of clusters in the whole leaf nodes must be less than M. To sum up, the tree which satisfies the above conditions can be called Cluster Feature tree (CF tree).

NCF tree: NCF tree is a CF tree which is composed of CFs derived from normal behaviors. NCF tree represents user's profiles of normal behaviors.

ACF tree: ACF tree is a CF tree which is composed of CFs derived from abnormal behaviors. ACF tree represents the known type of attack.

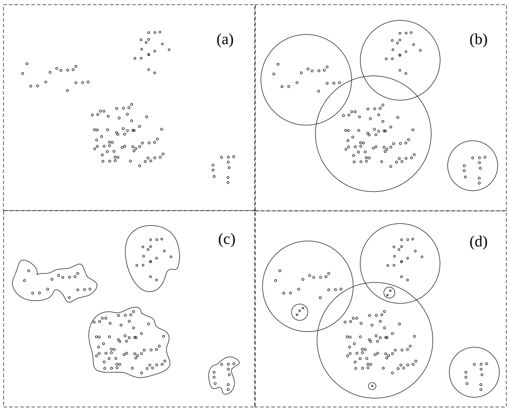

Figure 1. Clustering in three different ways

In the Fig 1, sub graph (a) shows some behavior points which needs to be clustered; sub graph (b) shows the hierarchical clustering result; sub graph (c) shows the density clustering result; sub graph (d) shows new clustering result after building NCF tree and ACF tree. The cluster shape of density clustering can be arbitrary. But the cluster shape of hierarchical clustering can only be spherical. So the accuracy of profiles which are generated by hierarchical clustering is not as good as profiles of density clustering. Therefore, as is shown in the sub graph (d) of Fig 1, ACF tree is built to dig the abnormal behaviors from the normal profiles.

III. PARALLEL ALGORITHM

The process of generating profiles and the process of predicting intrusion which are the bottlenecks of improving performance need a large number of calculation and run in the real time environment. Therefore, parallel design and implementation mainly focus on two processes, which are shown the Fig 2.

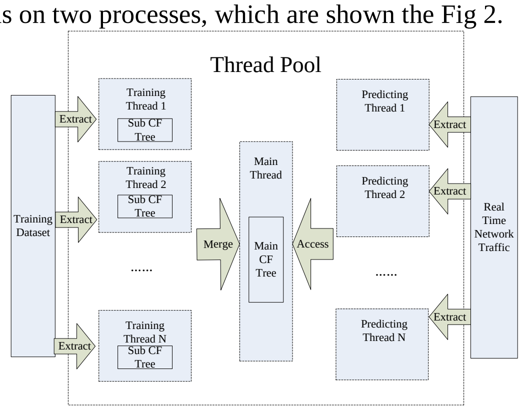

Figure 2. Overview of parallel algorithm

In the process of generating profiles, ACF tree is composed of abnormal behaviors which are added regularly by security administrator. So the real time amount of data is not too large. ACF tree can be built by a single thread instead of parallel. But NCF tree is composed of real time traffic. So it has to be built by multiple parallel threads.

The process of generating profiles includes two algorithms: parallel training algorithm and merging algorithm. The process of predicting intrusion includes an algorithm: parallel predicting algorithm. The three algorithms will be separately introduced as follows:

A. Parallel Training Algorithm

The working thread can not directly access the main thread. Moreover, main tree is running in the main thread. So there is no need of locking between working thread and main thread. Once main thread receives the sub CF tree, it will launch the merging algorithm and merge sub CF tree into main CF tree. Each thread can run in parallel with other threads and send sub CF tree to main thread.

The pseudocode of procedure WorkingThreadRun() is shown as follows:

```
Procedure WorkingThreadRun ()
{
        create a empty CF tree;
        while(new traffic){
          receive traffic(thread will sleep until the coming of traffic);
          extract v⃗ from traffic and generate CF;
          Insert(CF, root);//initial root is an empty CF
        if( CF tree is greater than M ){
            collect all the clusters of leaf nodes;
            send the above clusters to main thread;
            clean CF tree;
          }
        }
}
```

Initially, CF tree only has an empty root. With the coming of new vector $\vec{v}$, a temporary CF={1, $\vec{v}$, $\vec{v}^2$} is constructed, in other words, the vector $\vec{v}$ is considered as a cluster which only contains one vector. And the following vectors are dynamically inserted into the CF tree by procedure Insert(CF,p). The pseudocode of procedure Insert(CF,p) is described as follows:

```
Procedure Insert(CF, p)
{
        Traverse all the items of node p, find the nearest cluster CF_i;
        if( p is leaf node){
          if(merge CF into CF_i and new cluster still satisfies T)
            merge CF into CF_i //according to Theorem 1;
          else
            add a new item [CF, NULL] in node p;
        }else{
            insert(CF, pChildi); //recursive call
            merge CF into CF_i;
            if(pChild_i violate B){
              call SplitNode(pChild_i) //split pChild_i into pChild_1 and pChild_2;
              delete item[CF_i,pChild_i] from node p;
              create CF_1 and CF_2, they are separately cluster mergers of pChild_1 and pChild_2;
              create two new items [CF_1,pChild_1] and [CF_2,pChild_2];
              insert two new items into node p;
            }
        }
}
```

It can be drawn from the above pseudocode that the function of Insert(CF,P), as a recursion, needs to check whether the constraint B is satisfied or not when it returns. Therefore, the constraint B is split into two from bottom to top. If the root is against B, the tree would grow itself.

The above algorithm calls the sub procedure SplitNode(p), and the pseudocode shows details of SplitNode(p) as follows:

```
Procedure SplitNode(p)
{
        Traverse node p and find the two farthest nodes CF_i and CF_j;
        Generate two new nodes p1 and p2;
        Insert separately [CF_i,pChild_i] and [CF_j,pChild_j] into node p1 and node p2;
        for(all the CFs of node p except CF_i and CF_j){
          if( dis(CF,CF_i) ≤ dis(CF,CF_j) )
            insert [CF,pChild] into p1;
          else
            insert [CF,pChild] into p2;
        }
        return p1 and p2;
}
```

It can be drawn from the above algorithm that with the coming of new data, the number of leaf node increases until the constraint M is violated. Then CF tree needs to be rebuilt.

The process of rebuilding CF can be described as follows: first, the constraint T is relaxed. Then CFs of the entire leaf nodes are collected and CF tree is cleaning. Next the collected CFs are inserted into the new CF tree by calling the procedure of Insert(CF, p). Finally the CF tree is rebuilt.

Threshold T is one of three parameters of CF tree. To relax T can merge clusters and can reduce the number of leaf nodes. There are still two other parameters to be explained: B and M. B represents the maximum number of branches, in other words, node can have the maximum number of child nodes. Obviously, the larger B is, the lower the height of CF tree is. So if B is too small, the height of CF tree is high. And CF tree has a small number of CF. Such a CF tree can usually not satisfy the fundamental demand of clustering. If B is too large, the height of CF is low. A lager number of CFs will be gathered in few nodes. Such a CF tree usually leads to the performance problems of detection. Especially, the process of searching CF will traverse all the CFs in nodes. The real time of algorithm is difficult to guarantee.

B. Merging Algorithm

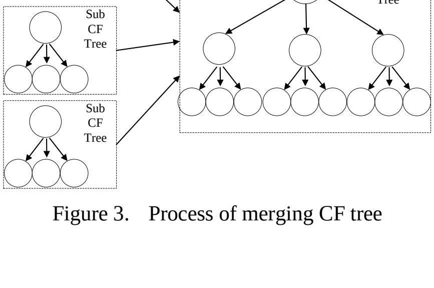

Figure 3. Process of merging CF tree

As shown in the Fig 3, the main thread receives sub CF tree and merge it into main CF tree. This process call the above procedure insert(CF, root).

The pseudocode of merging algorithm is described as follows:

```
Procedure MainThreadRun ()
{
        Create an empty CF tree;
        while(receive CF from working thread){
          for(each CF_i){
            Insert(CF_i, root); //root of main tree
            if(main tree violate M)
              relax T, rebuild main CF tree;
          }
        }
}
```

C. Parallel Predicting Algorithm

In the parallel predicting algorithm, multiple parallel predicting threads extracts $\vec{v}$ from the network traffic and predicts $\vec{v}$ with the help of profiles. Parallel predicting threads only read the profiles and do not write them. Multiple parallel predicting threads have no impact on CF tree. There is no mutual exclusion problem among predicting threads. Moreover, due to the nature of intrusion detection system, security administrator does not usually train and predict at the same time. As a result, multiple predicting threads and main thread hardly access the main CF tree at the same time. But in order to avoid this pitfall, main thread is protected by write lock and predicting threads are protected by read lock.

The predicting thread can be described as follows:

```
Procedure PredictThreadRun ()
{
        while(new traffic){
          receive new traffic //thread will sleep until the coming of traffic;
          extracts v from traffic;
          if(PredictACF (v ,root) is abnormal){
            alert abnormal;
          }else if(PredictNCF(v ,root) is abnormal){
            Alert abnormal;
          }
          Record normal;
        }
}
```

At first, PredictACF($\vec{v}$,**root**) is called to determine that $\vec{v}$ is the labeled attack type or not. If procedure of predictACF($\vec{v}$,**root**) directly returns abnormal; the detection of $\vec{v}$ is over. Otherwise, $\vec{v}$ needs to be further determined by calling **PredictNCF**($\vec{v}$,**root**). The details of procedure PredictACF($\vec{v}$,**root**) are shown as follows:

```
Procedure PredictACF (v , p)
{
        traverse all items of node p, and find the nearest CF_i of v ;
        if( p points to leaf node){
          if(the distance between v and CF_i < K*R(CF_i))  //k is a constant slightly greater than 1
            return labeled attack type of CF_i
          else
```

```
            return normal;
            PredictNCF(V ,root);
        } else
            return PredictACF(V ,pChild_i);
}
```

The details of procedure PredictNCF($\vec{V}$,**root**) are shown as follows:

```
Procedure PredictNCF(V , p)
{
        traverse all items of node p, and find the nearest CF_i of V ;
        if( p points to leaf node){
            if(the distance between V and CF_i > K*R(CF_i)) // k is a constant slightly greater than 1
                return abnormal;
            else
                return normal;
        } else
            return PredictNCF(V ,pChild_i);
}
```

The number of predicting threads can be dynamically adjusted according to the real time network traffic.

IV. Experimental Result and Analysis

A. Dataset

KDD CUP 1999 data set which is deprived from 1998 DARPA Intrusion Detection Evaluation program held by MIT Lincoln Labs, is employed to study the utilization of machine learning for intrusion detection by numerous researchers. The dataset includes all kinds of simulated intrusion actions in the complicated network environment, where each connection instance contains 41 features. In this paper the KDD CUP 1999 data set have been selected as the simulated traffic source of our experiments. 100,000 connection instances, as the training dataset, are extracted randomly from the file kddcup. data_10_percent. 100,000 connection instances, as the predicting dataset, are extracted randomly from the file corrected.

B. Optimal Parameter

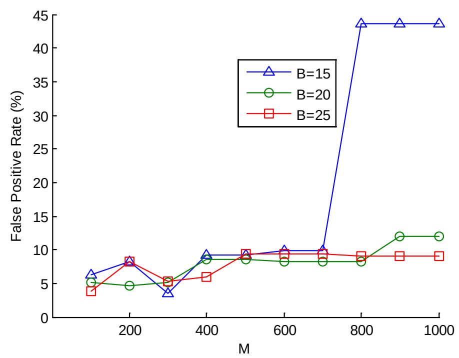

Figure 4. Optimal B and M (false positive rate)

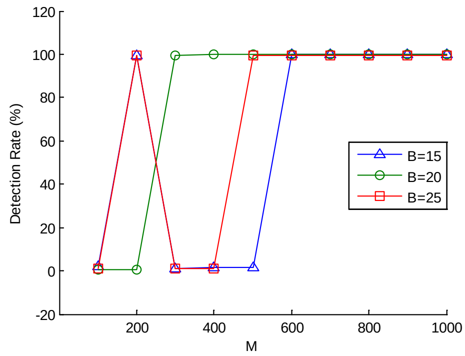

Figure 5. Optimal B and M (detection rate)

There is essentially no difference between serial execution result and parallel execution result. So the optimal parameters of main CF tree are selected by referring to the optimal parameters of serial procedure. As is shown in the Fig 4 and Fig 5, only when B=20 and M=300, detection rate is relatively high and false positive rate is relatively low in the tolerable range.

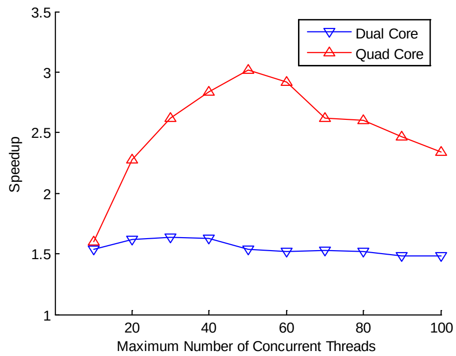

Figure 6. Optimal number of concurrent threads

Threshold T_M of main CF tree can relax itself when the number of clusters is great than M. Due to relaxing T, some clusters are merged and the number of leaf nodes decreases. The initial value of T_M of main CF tree is set to 0, and T_M will reach an adaptive value with the growth of CFs. The final T_M is equal to 0.8.

Every CF tree has its parameters. But sub CF tree has no direct impact on detection performance. So there is no need to select the optimal parameters for sub CF tree. The parameters of sub CF tree still need to take some basic precautions: (1) B of sub CF tree should be same with B of main CF tree. It helps to be convenient to insert sub CF tree into main CF tree; (2) M of sub CF tree should be less than M of main CF tree, which can reduce times of splitting node from bottom to up; (3) T_S of sub CF tree should be a constant which is less than T_S of main CF tree. Therefore the parameters of sub CF tree are set as follows: B=15, M=200 and T_S=0.5.

The maximum number of concurrent threads in thread pool has various default values according to different CPU. This value can be modified according to the actual need. The maximum number of concurrent threads in thread pool should be increased with the growth of real time network traffic. But too many threads in thread pool will also lead to the incremental communication overhead between threads and the incremental concurrency control cost. So the optimal number of threads in thread pool should be selected.

As is shown in the Fig 6, parallel predicting procedure which runs in the Dual Core reaches the maximum speedup when the maximum number of threads is equal to 30 and the parallel predicting procedure which runs in the Quad Core reaches the maximum speedup when the maximum number of threads is equal to 50.

C. Performance Comparison

There are three indexes for measuring performance: Speedup, detection rate and false positive rate.

Speed up is the ratio of running time T_{sp} of serial procedure to running time T_{pp} of parallel procedure when the core number of processor is n:

$$S_n = T_1 / T_n \qquad (4)$$

Different sizes of network traffic are simulated by different number of connection instances. The basic executable unit of procedure is a connection instance. The serial procedure can only process one connection instance at a time, and the parallel procedure can process multiple connection instances. The number of processing connection instances is determined by the number of idle threads in thread pool.

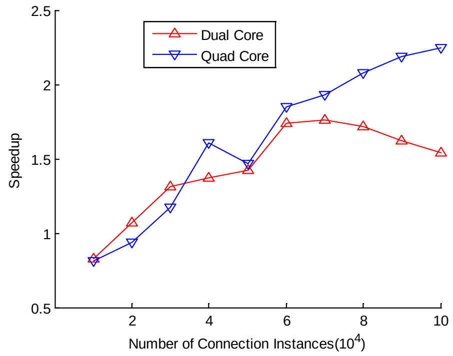

Figure 7. Speedup of parallel training procedure

As shown in the Fig 7, with the increasing number of connection instances, the speedup of parallel training algorithm which runs in the Dual Core processor reaches the maximum when the number of connection instances is equal to 60000. Then its speedup decreases slightly. And the speedup of Quad Core has continued to rise. It can be drawn that the parallel training algorithm running in Quad Core processor has the best real time performance in the face of a large amount of network traffic. Moreover, it is noticeable that when the number of connection instances is small, parallel procedure spent more time than the serial procedure.

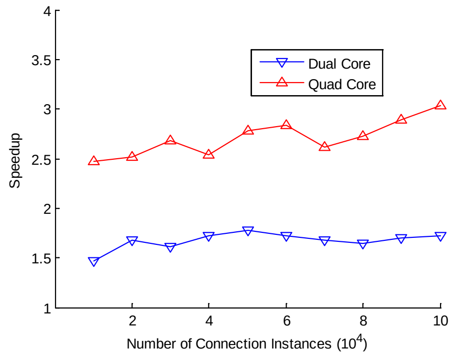

Figure 8. Speedup of parallel predicting procedure

Fig 8 shows the speedup of parallel predicting algorithm which runs in the Dual Core processor and Quad Core processor. Compared with speedup of parallel training algorithm, parallel predicting algorithm has higher speedup when the number of connection instances is small. And the number of connection instances has smaller impact on parallel predicting algorithm running in the Dual Core processor than Quad Core processor.

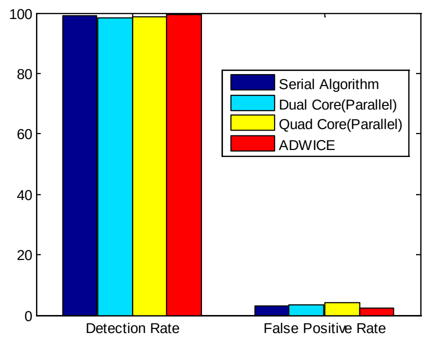

Figure 9. Comparison of detection performance

It can be drawn from Fig 9 that serial procedure or parallel procedure, neither side has any significant advantage on the detection performance. The detection performance of parallel anomaly detection algorithm is close to the adaptive anomaly detection algorithm ADWICE [15].

V. CONCLUSION

With the advantage of multi-core, two busiest processes in the detection: training and predicting are split into many sub tasks and are carried out at the same time instead of one after the other. The parallel processing of training and predicting has the same excellent detection performance with serial processing, and it also has better real time performance than serial processing. Moreover, ACF tree can compensates for the loss of profiles accuracy which is derived from the spherical cluster.

In the future, the message communication mechanism between threads will be replaced by the network communication mechanisms. Parallel threads will run in different hosts. We will focus on a distributed anomaly detection algorithm.

ACKNOWLEDGMENT

This work was supported in part by the National High Technology Research and Development Program of China under Grant No. 2011AA010101, the National Natural Science Foundation of China under Grant No. 61103197 and 61073009, the Key Programs for Science and Technology Development of Jilin Province of China under Grant No. 2011ZDGG007, the Youth Foundation of Jilin Province of China under Grant No. 201101035.

REFERENCES

[1] Erbacher RF, Walker KL and Frincke DA, "Intrusion and misuse detection in large-scale systems," IEEE Computer Graphics and Applications. vol: 22, pp: 38-47, January 2002.

[2] Depren O, Topallar M and Anarim E, "An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks," Expert Systems with Applications. vol: 29 pp: 713-722, November 2005.

[3] Zhan Justin, Oommen B. John and Crisostomo Johanna, "Anomaly Detection in Dynamic Systems Using Weak Estimators," ACM Transacctions on Internet Technology. vol: 11, July 2011.

[4] Khazai Safa, Homayouni Saeid and Safari Abdolreza, "Anomaly Detection in Hyperspectral Images Based on an Adaptive Support Vector Method," IEEE Geoscience and Remote Sensing Letters. vol: 8 pp: 646-650, July 2011.

[5] Androulidakis Georgios and Papavassiliou Symeon, "Two-stage selective sampling for anomaly detection: analysis and evaluatio, " Securiyt and Communiation Networks. vol: 4, pp: 608-621, June 2011.

[6] Patcha Animesh and Park Jung-Min, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," Computer Networks. vol: 51, pp: 3448-3470, August 2007.

[7] Thottan M and Ji C, "Anomaly detection in IP networks," IEEE Transactions on Signal Processing. vol: 51 pp: 2191-2204, Aug 2003.

[8] Ester M. and Kriegel H.-p. "A density –based Algorithm for discovering clusters in large spatial databases with noise," Proc. 2nd Int. Conf. on knowledge discovery and Data mining. Portland USA, pp 226-231, 1996.

[9] Derya. Briant and Alp. Kut. "ST-DBSCAN: An algorithm for clustering spatial-temporal data,". Data & Knowledge Engineering. vol: 60, pp:208-221, January 2007.

[10] Tran Thanh N., Wehrens Ron and Buydens Lutgarde M. C, "KNN-kernel density-based clustering for high-dimensional multivariate data," Computational Statistics & Data Analysis," vol: 51, pp: 513-525, November 2006.

[11] Brecheisen S, Kriegel HP and Pfeifle M, "Multi-step density-based clustering," Knowledge and Information Systems. vol: 9, pp: 284-308, March 2006.

[12] Karypis G, Han EH and Kumar V, "Chameleon: Hierarchical clustering using dynamic modeling," Computer, vol: 32, pp: 68, August 1999.

[13] Tsekouras G, Sarimveis H and Kavakli E; "A hierarchical fuzzy-clustering approach to fuzzy modeling," Fuzzy Sets and Systems. vol: 150, pp: 245-266, March 2005.

[14] Zhang T, Ramakrishnan R and Livny M. BIRCH: "an efferencient data clustering method for very large database," SIGOD record 1996 ACM SIGMOD international conference on management of data, vol:25(2), pp104-114, 1996.

[15] Buibeck K and Simmin Nadim-Tehrani. "ADWICE-anomaly detection with fast incremental clustering," In: Proceedings of the seventh international conference on security and cryptology, Springer Verlag, 2004.

[16] Murty R and Okunbor D, "Efficient parallel algorithms for molecular dynamics simulations," Ararllel Computing. vol: 25 pp: 217-230, March 1999.

[17] Xu XW, Jager J and Kriegel HP, "A fast parallel clustering algorithm for large spatial databases," Data Mining and Knowledge Discovery." vol: 3, pp: 263-290, September 1999.

[18] Rajasekaran S, "Efficient parallel hierarchical clustering algorithms," IEEE Transactions on Parallel and Distributed Systems. vol: 16, pp: 497-502, June 2005.

[19] Dursun Hikmet, Kunaseth Manaschai, "Hierarchical parallization and optimization of high-order stencil computations on multicore clusters." Vol 62, pp:946-966, Nov 2012.

[20] Chan Siew yin, Ling Teck Chaw, Aubanel Eric, "The impact of heterogeneous multi-core clusters on graph partitioning: an empirical study." Vol 15, pp:281-302, Sep 2012

[21] Huang Lei, Wang Jiabing and He Xing, "A graph clustering algorithm providing scalability," Journal of Network, Vol 7, pp: 229-335, 2012.

[22] Chen Huimin, Bart Jr. and Henry L. and Huang Shuqing, "Integrated feature selection and clustering for taxonomic problems within fish species complexes," Journal of Multimedia, Vol 3, pp: 10-17, July 2008.

[23] Zhang Qiaorong, Zheng Yafeng, Liu Haibo, Shen Jing and Gu Guochang, "Perceptual object extraction based on saliency and clustering," Journal of Multimedia, Vol 5, pp: 393-400, 2010.

Wei-wu Ren studies in the College of Computer Science and Technology at Jilin University, and separately gained a bachelor's degree in 2007 and a master's degree in 2010. Now, he is a doctorate candidate in the same university. His research interests are information security, data mining and knowledge representation.

Liang Hu received his Ph.D. degree in Computer Software and Theory from Jilin University in 1999. He is currently a professor in the College of Computer Science and Technology, Jilin University. His research interest covers network security and distributed computing, including related theories, models, and algorithms of PKI/IBE, IDS/IPS, and grid computing.

Kuo Zhao received the B.E degree in Computer Software in 2001 form Jilin University, followed by M.S degree in Computer Architecture in 2004 and Ph.D. in Computer Software and Theory from the same university in 2008. He is currently associate professor in the College of Computer Science and Technology, Jilin University. His research interests are in operating systems, computer networks and information security. He is the corresponding author of this paper.

Jianfeng Chu (chujf@jlu.edu.cn) ***corresponding author*** is currently an associate professor in the College of Computer Science and Technology, Jilin University. He received his Ph.D. degree from Jilin University. His research interests include network security, mobile security, cloud security and security of Internet of Things.