IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 13, 2019, accepted August 22, 2019, date of publication August 28, 2019, date of current version September 13, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2938044

Using Cache Optimization Method to Reduce Network Traffic in Communication Systems Based on Cloud Computing

LANLAN KANG[1], RUEY-SHUN CHEN[2], YEH-CHENG CHEN[3], CHUNG-CHEI WANG[4], XINGGUAN LI[5], AND TSU-YANG WU[6]

[1]College of Applied Science, Jiangxi University of Science and Technology, Ganzhou 214322, China
[2]Department of Computer Engineering, Dongguan Polytechnic, Dongguan 523808, China
[3]Department of Computer Science, University of California, Davis, CA 95616, USA
[4]Institute of Information Management, National Chiao Tung University, Hsinchu 30050, Taiwan
[5]School of Mathematics and Statistics, Minnan Normal University, Zhangzhou 363000, China
[6]College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China

Corresponding author: Tsu-Yang Wu (wutsuyang@gmail.com)

The work of L. Kang was supported in part by the Natural Science Foundation of China under Grant 61170305 and Grant 11704164 and in part by the Ph.D. Research Startup Foundation of Applied Science College of Jiangxi University of Science and Technology under Grant YJ2018-02. The work of R.-S. Chen was supported in part by the Scientific Research Fund of Dongguan Polytechnic under Grant 2018a02.

ABSTRACT The overwhelming increase of population will lead to an increase network traffic usage. In most products, local cache mechanism is implemented for the purpose of reducing the network traffics. However, Due to storage space limitations, lot of times, cache will get purged when there are lots of queries sent and replaced by other newest queries. It is not an efficient by using traditional methodology to build or forming cache. In order to reduce the network traffic usage, the paper propose a solution, which utilizes data mining technique with clustering concept, by gathering the current feedback data that have from our SPN (Smart Protection Network). we are able to form these data in groups with similarity, and by deploying these data to client side, to achieve the reduction of traffic usage. In prototype, this design can really reduce network traffic more than 8%.

INDEX TERMS Cloud computing, cache optimization, network traffic, data mining.

I. INTRODUCTION

The cathe optimization method with innovation technology over the cloud technology [4], [7]–[10], [45]–[58], or known as the SPN (Smart Protection Network). The Smart Protection Network infrastructure delivers advanced cloud protection and block threats in real-time before they reach you. Smart Protection Network delivers security that's smarter than conventional approaches. Leveraging cloud computing [5], [6], [44] across the company's security solutions and services, the Smart Protection Network provides a stronger cloud architecture that protects you while reducing your reliance on time-consuming signature-downloads.

Although using the Content Distributed Service provided by CDN (Content Delivery Network) [1], [2] company, the SPN works for all over the world. But, for the Security solution provider, the cloud server to serve SPN service is limited, the charge by CDN company also based on the network traffic; for client side, using huge network traffic is not permitted after install the SPN service for network security [36]–[43]. It is clear to reduce that the overloading of network traffic usage will eventually become a major issue for those products which adopt the SPN solution.

At the same time, though, the Internet network traffic being handled by the backend servers of a file reputation service is increasing dramatically in terms of incoming queries. Network traffic is increasing week-by-week and this means many thousands of dollars must be paid by the operator of the file reputation service to an Internet traffic company such as Akamai [3] for the increase in traffic.

In the field of antivirus protection for computers and enterprise networks, a file reputation service may be employed to check the legitimacy of certain files on user computers. Current file reputation services (or, file reputation technology) often use a cache [10]–[17] mechanism to speed up response times. A file reputation service checks the reputation of a file against an extensive in-the-cloud database before permitting user access. The service uses

The associate editor coordinating the review of this article and approving it for publication was Raja Wasim Ahmad.

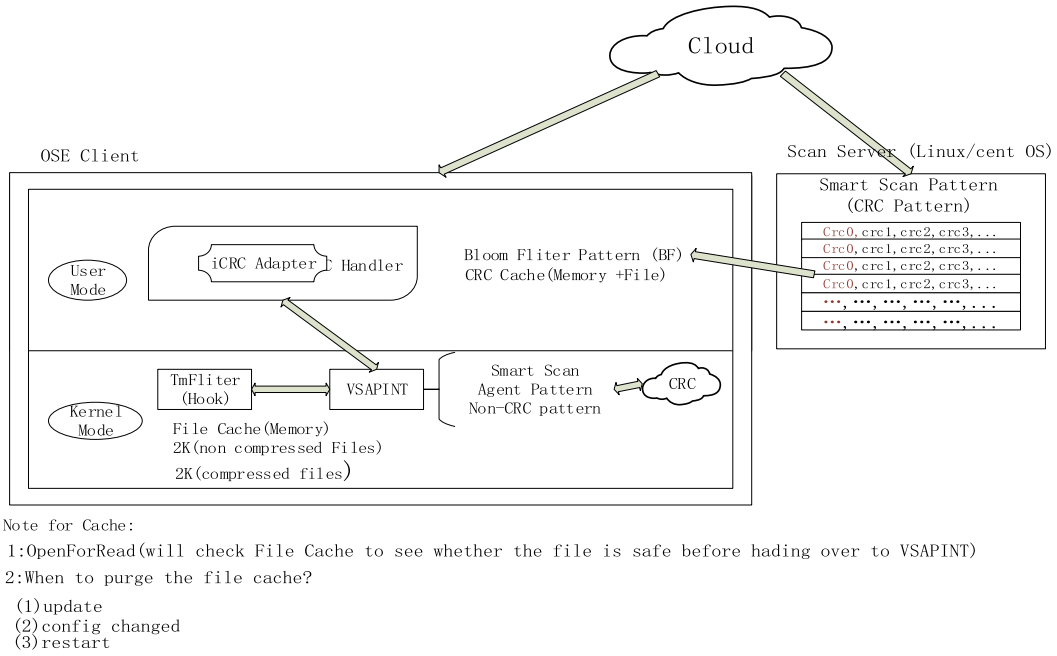


FIGURE 1. iCRC scan.

high-performance content delivery networks and local caching servers in order to assure minimum latency during the checking process. Typically, the cache stored on the local, client-side is based upon queries sent to the file reputation service. But, this cache may be purged if many queries are sent or if there is an incomplete update; this presents a problem in that the cache is then not useful. So, this paper tries to do ''cache optimization'' based on the cache contents.

The CDN using Akamai as an example (there are other such services) in the context of a file reputation service, the service may upload data to one Akamai server. Then, the Akamai server will distribute this data worldwide to Akamai servers. Finally, clients can access the nearest Akamai server to download what they need. Of course, all of this activity means increased network traffic and cost.

Network traffic usage is increasing not only more clients using the service, but also a limitation on the local client cache which requires a query to be sent out over the Internet to the backend servers. As mentioned above, this limitation is that the cache might be purged prematurely, and there is no quick and easy way to find out which items in cache should be kept without downloading them again and again.

Because a response from a backend server of the file reputation service may take more than 350 ms, while a response from a local client cache may take only 1ms, it is important to make better use of a client cache instead of relying upon access to a backend server. To those ends, further techniques for better utilizing a local client cache in the context of a file reputation service or for any other suitable service are desired.

An access log of a backend server includes any number of data records each describing a previous query regarding a suspect computer file of a client computer. Each record includes the CRC code for the suspect computer file, the result of the malware analysis performed on the backend server and other attributes and values. The log is analyzed to retrieve relevant attributes and values from each record. Key attributes and values are generated from the data in each data record such as region, continuous query, up-to-date pattern and CRC summary count. All CRC codes are grouped according to attribute values that each has in common with one another. There may be many groups formed each including any number of CRC code-result pairs. Each group is analyzed to determine the network traffic associated with downloading the entire group to all user computers and the network traffic associated with not downloading the group but responding to future malware queries regarding CRC codes in the group. CRC codes are removed from each group if necessary and the network traffic is iteratively calculated to maximize the reduction in network traffic. Each group may be then downloaded to all user computers as a pre-fetch cache.

To find out the perfect pre-fetch cache for clients to reduce network traffic. We present a technique is disclosed that allows a pre-fetch cache to be downloaded all at once to a client computer.

Whereas a client with a traditional cache needs to send out multiple queries in order to accumulate a usable cache one-by-one, the present invention provides a customized pre-fetch cache generated in the cloud that the client may retrieve with on the order of only a few queries. In other words, the entire pre-fetch cache can essentially be retrieved at once using the steps described below.

II. RELATED RESEARCH

A. iCRC [8] SCAN

The following Fig. explain what the iCRC scan do between client and cloud servers. CRC [14] is a term stands for cyclic

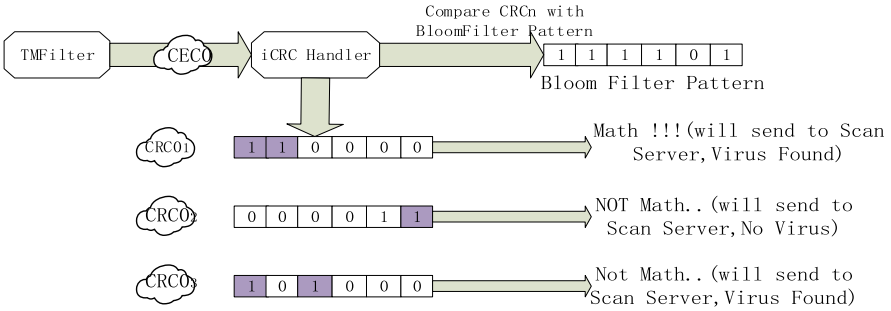


FIGURE 2. Compare CRC with bloom filter pattern.

redundancy check. A Cyclic Redundancy Check (CRC) is an error-detecting code designed to detect accidental changes to raw computer data, and is commonly used in digital networks [37]–[39], [42] and storage devices such as hard disk drives. CRCs are so called because the check (data verification) code is a redundancy (it adds zero information to the message) and the algorithm is based on cyclic codes. CRCs are popular because they are simple to implement in binary hardware, are easy to analyze mathematically, and are particularly good at detecting common errors caused by noise in transmission channels. As the check value has a fixed length, the function that generates it is occasionally used as a hash function. The CRC was invented by W. Wesley Peterson in 1961; the 32-bit polynomial used in the CRC function of Ethernet and many other standard is the work of several researchers and was published in 1975.

The company uses the idea of Bloom Filter [16] as the CRC checking rule. And then form the Bloom Filter Pattern to do pre-scan during file scanning.

iCRC Scan Flow:

(1). TMFilter hooks a file.

(2). TMFilter looks up file cache [11,12] to see if this file has been scanned before.

(3) If YES, finds out & returns last scan result.

(4) If NOT, passes the file to VSAPINT.

(5). VSAPINT calculates CRC checksum for this file.

(6). VSAPINT passes CRC checksum value to iCRC Handler.

(7). iCRC Handler looks up Bloom Filter to see if there is a match for this CRC value.

(8)If YES, proceeds to see if corresponding CRC result is already in CRC cache; otherwise make further query to Scan Server.

(9)If NOT, returns No Virus (File not malicious).

(10). (The product OSCE will call) VSAPINT to do the follow-up virus info query (virus name & action to take) if the CRC match determines this file is malicious.

For clients that use Global Scan Server, the step 5 queries go to CLOUD will hit Akamai first to see if the same CRC query result has been cached; otherwise get CRC result directly from Global Scan Service.

B. FILE REPUTATION SERVICE (FRS)

In addition to Web and e-mail reputation technologies, traditional malware scanning tools identify infected files by comparing several hash values of the file content with a list of hash values stored in a pattern or signature file. If a file is marked "suspect" in the first pass of a hash comparison, the scan engine employs a multi-phase approach to further drill down. In many conventional tools, this pattern file is located on the endpoint computer and is distributed regularly to provide protection against the latest threats.

File reputation technology decouples the pattern file from the local scan engine and conducts pattern file lookups over a network to a file reputation service server. The server may reside on the customer premises (a private cloud) or even on the Internet (a public cloud). This in-the-cloud approach alleviates the challenge of deploying a large number of pattern files to hundreds or even thousands of endpoints. With this approach, as soon as the pattern is updated on the cloud server, protection is immediately available to all clients accessing that scan server. The file reputation databases in the cloud receive constant updates—enabling a provider to quickly respond to and remediate new file-based threats. Similar technologies are also available for Web and e-mail scanning.

FIGURE 3. File reputation technology.

Fig. 3 illustrates a prior art file reputation service. Shown are any number of user computers and their respective files that need checking for malicious content. An individual file may be sent over the Internet to a file reputation service where

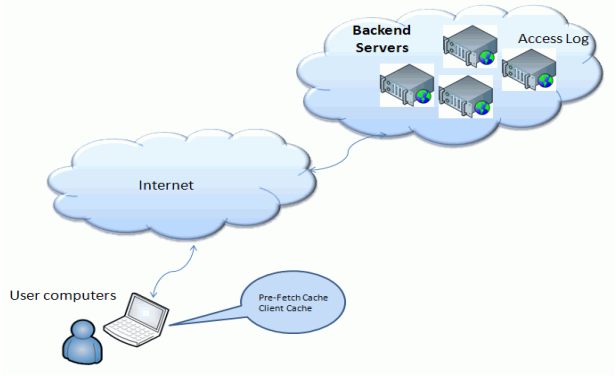


FIGURE 4. SPN mechanism.

a database is used to check the contents of a file to determine whether or not it is malicious.

C. SPN MECHANISM

Fig. 4 is a block diagram illustrating one embodiment of the invention. Shown are any numbers of user computers that communicate over the Internet or other network to any number of backend servers which are typically operated by an entity such as an antivirus service provider. These user computers may represent an individual computer in a household that is running an antivirus software product, an office computer within a corporation running an antivirus business software product, a server computer within an entity (being responsible for the security of any number of end-user computers) running an enterprise version of a security software product, or other computer arranged to submit queries and receive results as described herein. These user computers are preferably distributed worldwide within any number of independent entities such that the queries received from them best represents a worldwide or region-wide view of malware activity.

The antivirus service provider may use a single backend server to field queries and provide results, or may use any number of backend servers to process the queries. For example, a particular backend server may be dedicated to a certain product or products of an entity may be used to respond to queries from particular countries or regions for load balancing purposes, or the provider may pay a Web service provider (e.g., Akamai) to provide this kind of service.

As described above, a user computer determines that a file is suspect, calculates its CRC and sends this CRC code as a query to one of the backend servers. There may be more than one CRC code calculated for an individual file that must be checked. Typically, a user computer checks a CRC code against a security software product, various lists or a local cache (e.g., client cache) before making the determination that the CRC is suspect and must be sent over the Internet. Not every calculated CRC must be sent over the Internet to the backend servers, but if it is, this increases the network traffic and the associated burden on the backend servers and cost to the antivirus service provider. One of the backend servers will receive this query, determine whether the CRC code represents a malicious file (using any combination of well-known or proprietary techniques) and send a result back to the requesting user computer. This result typically includes the original CRC and a result value indicating whether or not the computer file from which the CRC was calculated is malicious or not.

The present invention realizes that it is possible to deliver a pre-fetch cache to each of the user computers against which each user computer may compare a suspect CRC code. Advantageously, this pre-fetch cache need not be built up query-by-query as in the prior art and may be delivered to each user computer periodically with a minimum of effort expended by the user computer and with a minimum impact on network bandwidth. Accordingly, a user computer is supplied with a relatively large pre-fetch cache (compared to the prior art client cache) that is available for use and will dramatically cut down on the number of queries needing to be sent to the backend servers. Any user computer may use the pre-fetch cache on its own or in conjunction with the client cache. As with the client cache, the pre-fetch cache typically will include a number of CRC code-result pairs.

An access log is a database including any number of records where each record reflects a query sent from a user computer around the world. There may be a single access log representing any number of backend servers or each backend server may keep its own access log (not shown). In the case where each backend server has its own access log, a single access log may be developed by combining all of the individual access logs. In the example shown, backend servers deliver records from queries unidirectional to the access log, and in addition, backend server is able to retrieve records from the access log in order to implement the present invention using suitable software. The pre-fetch cache developed upon server will eventually be deployed to each of the user computers desiring to implement the present invention.

D. TWOSTEP CLUSTERING

Using PASW(SPSS) 17.0 to conduct a cluster analysis [18]–[20].

The SPSS two-step clustering component is a scalable clustering analysis and the MapReduce mode [26]–[29], [31]–[35], [40], [41] algorithm for processing very large data sets. Ability to handle continuous and categorical variables and attributes, it only needs to pass a single data in the process. In the first step of the process, you pre-aggregate records into many small sub-clusters. Then, cluster the sub-cluster from the pre-clustering step to the number of clusters required. If the number of clusters required is unknown, the SPSS two-step cluster component will automatically find the appropriate number of clusters.

III. ARCHITECTURE DESIGN

The solution begin with analyzing current data we have from the file reputation query, follow by running a statistical

method-two step clustering and forming different groups with similarity from the output of Chi-Square distribution analysis [21], [22],and finally adopt mathematical algorithm calculation to produce the outcome for deployment common cache. Based on the improvement of cache contents, here comes the useful pre-fetch cache, so we called it's a kind of "Cache Optimization" [6], [41].

To explain these steps in details, we have transformed this solution into a repeatable process, and broken down the process into six steps, which is described in detail below.

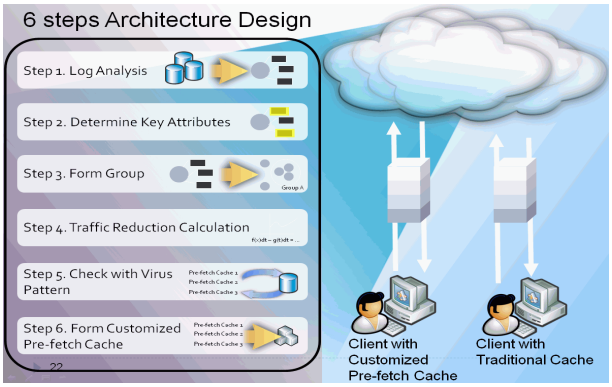


FIGURE 5. Six steps architecture design.

Fig. 5 is a high-level flow diagram describing one embodiment for implementing the present invention. It is contemplated that a suitable software module upon one of the backend servers will implement the steps shown in Fig. 5, although it is possible for the steps to be distributed among many computers and coordinated by a single computer. After an access log is obtained, located or accessed by a software module executing upon backend server. As mentioned above, this access log includes real-world data from around the world and from many different entities, documenting the CRC queries that have been sent to the backend servers for analysis. Each query results in a record being created and stored in the access log. For example, there may be over ten million records in the access log. Typically, a query on a user computer originates with a security software product that desires to know whether a particular file is malicious or not; the access log is typically also under control of the entity that sells the security software product. Many security software products that may send queries or other products from other anti-virus service providers.

A. ANALYZE ACCESS LOG ANALYSIS

In step 1 the access log is analyzed to obtain a variety of attributes and their values for each record, as well as defining additional attributes and values not necessarily present within the access log. In one embodiment, the method processes each record of the access log in turn in order to pull out particular fields and data; other information may be present within each record.

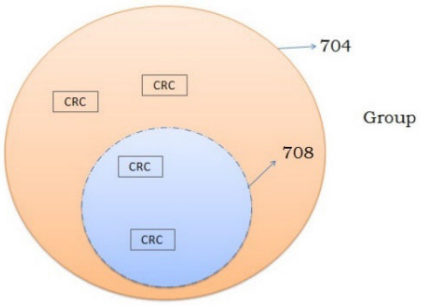


FIGURE 6. CRC group example 2.

B. DETERMINE KEY ATTRIBUTES

In step 2, certain key attributes of the retrieved and defined attributes of step 1 are identified for further use in the invention. These key attributes will then be used in the next step as characteristics to form groups of related CRCs. In other words, CRCs having the same values for these key attributes would be placed into a single group. Statistical tools may be used to determine the preferred key attributes. Since CRCs in the same group have similar behaviors (as evidenced by the key attributes having the same values), an assumption can be made that the client computers responsible for sending these CRC queries may have similar behaviors and one may predict future user queries from these client computers. As explained below, a unique pre-fetch cache may be generated for client computers in the same group. In this fashion, we maximize the hit rates of queries on the pre-fetch cache to reduce queries over the Internet to the backend servers and reduce costs.

C. FORM GROUPS OF RELATED CRC ENTRIES

After each key attribute is defined from step 5, we then form the CRC into group with similarity. Examples are like CRCGroupA contains the attributes of (Product:OSCE, UpToDatePattern:Yes,ServerGUID:Query:1).

Once the key attributes have been determined in step 3 any numbers of groups are formed of related CRC entries. Preferably, groups contain CRC-result pairs, i.e., each item in a group is a CRC code paired with its result (malware or not malware). It is possible that some groups may not have any items.

Now that the groups of related CRC entries have been formed it is possible to deploy these groups as pre-fetch caches to the various user computers. For example, the group Y formed above may be deployed to all client computers in the North America region because this group has the attribute North America region in common (among others), and it is likely that other client computers in this region would be attempting to send the same CRC query as other computers in this region. In another example, all of the CRCs of the group A formed above may be sent to all client computers in Taiwan. In a further refinement, it may be desirable to only send this group to computers in Taiwan using the particular product that they have in common.

Because the size of each of these groups of CRC entries may be quite large, though, it is beneficial to perform a traffic reduction calculation first in order to determine whether deploying these groups actually will reduce network traffic.

D. TRAFFIC REDUCTION CALCULATION

In the described embodiment, because there are over eight million CRC entries formed into 99 groups, it is possible that some of the groups may be on the order of thousands or hundreds of thousands of entries in size. Downloading one or some of these groups as a pre-fetch cache to a single client on a user computer may actually take up more network traffic bandwidth than if the present invention were not implemented and the user computer built up its cache one-by-one as queries were submitted. Therefore, it is beneficial to perform a calculation of traffic volumes and even to determine if certain CRC entries should be deleted from a group before that group is downloaded to a client as a pre-fetch cache. This calculation is performed on each group.

The traffic reduction calculation may be performed on the basis of downloading a group to a single user computer or on the basis of downloading a group to all of the user computers. For example, assuming we have three groups where the calculation is positive, then we attempt to enforce that all user computers in these three groups download their dedicated pre-fetch cache. Preferably, we generate a different pre-fetch cache for each different group based on the calculation.

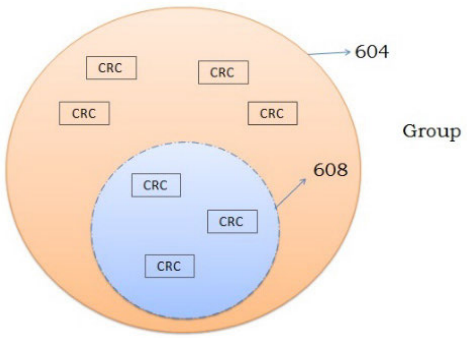


FIGURE 7. CRC group example 1.

Fig. 7 illustrates an example in which a group 604 includes any number of CRC entries shown inside the solid circle. If the group were downloaded all at once to a user computer as a pre-fetch cache in accordance with the present invention than all of the CRC entries within circle 604 would be downloaded and would contribute to a certain level of network traffic volume. But, if the present invention is not implemented, then all of these entries would not have to be downloaded at once but there is a certain subset 608 (shown in the dashed line) of CRC entries that would be queried by the user computer and results would have to be returned, also contributing to a certain level of network traffic volume. If the volume of network traffic created by downloading the pre-fetch cache 604 is greater than the volume of network traffic created in a forecast future time by responding to queries for subset 608, then it would not seem worthwhile to download group 604 as a pre-fetch cache.

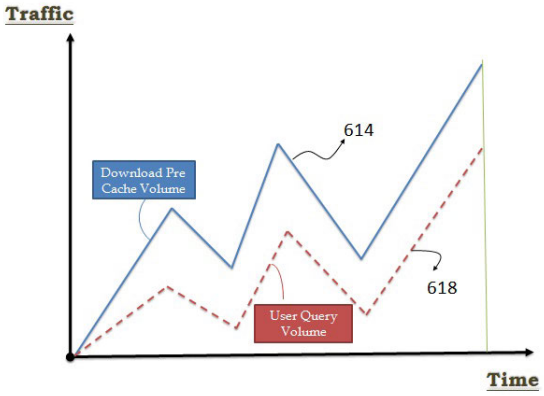


FIGURE 8. Comparison 1 of network traffic volumes.

Fig. 8 graphically illustrates a comparison of network traffic volumes created by either scenario. The x-axis represents a given future time period, such as the next seven days, while the y-axis represents the amount of traffic volume at a particular point in time. Therefore, if the amount of user queries can be forecast for the next seven days (i.e., normal queries resulting without implementing the present invention), then the area under dashed line 618 represents the user query volume in terms of network traffic bandwidth that is used up. In other words, a prior art approach in which only CRCs in subset 608 are queried over a future time period results in a user query volume. On the other hand, if the present invention is implemented then all CRCs in group 604 are downloaded all at once and the volume of the network traffic bandwidth used up may be estimated using the area under solid line 614. In this example, since the volume of network traffic created using the pre-fetch cache approach (area under line 614) is greater than the user query volume traffic created without deploying the pre-fetch cache (area under dashed line 618), then it is advisable not to deploy this group as a pre-fetch cache.

In this situation, it is possible to remove a certain number of CRC entries from group 604 and to perform the calculations again. If the CRC entries are removed intelligently (i.e., using criteria designed to reduce the volume associated with downloading the pre-fetch cache), then at some point the calculations may reveal that the volume associated with the pre-fetch cache is less than the volume associated with traditional user queries. This process of performing the calculations, removing CRC entries from the group, and performing the calculations again, is performed repeatedly on each group until a maximum possible savings in network traffic bandwidth is determined. Of course, it is possible that the difference between the two volumes for a particular group would always be negative (such as is shown in Fig. 7) in which case that particular group would not be downloaded as a pre-fetch cache.

Fig. 6 illustrates an example in which group 704 includes a reduced number of CRC entries shown inside the solid circle. A number of CRC entries have been removed from this group in order to determine if implementing the present invention is advantageous in this new situation. Removal of CRC entries from group 704 may result in removal of entries from subset 708 but not necessarily. In this new situation, if the volume of network traffic created by downloading the pre-fetch cache 704 is less than the volume of network traffic created in a forecast future time by responding to queries for subset 708, then it would seem worthwhile to download group 704 as a pre-fetch cache.

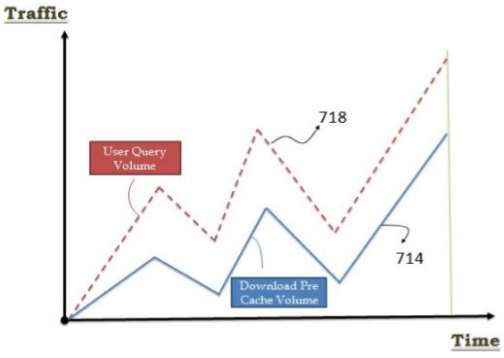


FIGURE 9. Comparison 2 of network traffic volumes.

Fig. 9 graphically illustrates a comparison of network traffic volumes created by this new scenario. The x-axis represents a given future time period, such as the next seven days, while the y-axis represents the amount of traffic volume at a particular point in time. Therefore, if the amount of user queries can be forecast for the next seven days (i.e., normal queries resulting without implementing the present invention), then the area under dashed line 718 represents the user query volume in terms of network traffic bandwidth that is used up. In other words, a prior art approach in which only CRCs in subset 708 are queried over a future time period results in a user query volume. On the other hand, if the present invention is implemented then all CRCs in group 704 are downloaded all at once and the volume of the network traffic bandwidth used up may be estimated using the area under solid line 714. In this example, since the volume of network traffic created using the pre-fetch cache approach is less than the user query volume traffic created without deploying the pre-fetch cache (area under dashed line 718), then it is advisable to deploy this group as a pre-fetch cache. Another way to reach this conclusion is to subtract volume 714 from volume 718 and if the result is positive, then the pre-fetch cache should be deployed.

Although Fig. 10 and 9 are presented as an iteration of the original group shown in Fig. 6 and 7, it is entirely possible that the initial calculation of a group may result in the scenarios shown in Fig. 12and 13. Even in this situation, it is advantageous to iterate over the group, strategically removing CRC

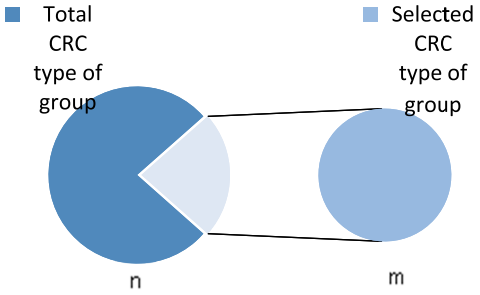


FIGURE 10. CRC group selection.

entries in order to achieve the maximum traffic bandwidth savings.

One particular embodiment of the present invention utilizes the below algorithm to calculate the estimated traffic reduction for each group (essentially subtracting the traffic volume associated with the pre-fetch cache from the traffic volume associated with traditional user queries). A positive value for the traffic reduction means the group should be deployed as a pre-fetch cache. After grouping all the CRCs from the previous step, this traffic reduction forecast algorithm will help to determine if a group is worthwhile for deployment as well what is the best deployment size for that group (i.e., should entries be removed before deployment).

Total CRC type of group = n

Selected CRC type of group = m

Consider that variable n = the total number of original CRC entries in a group, and that variable m = the number of CRC entries in a group once some entries have been removed. The algorithm then operates to count iterations of m from 0 to n, for the purpose of obtaining the maximum value of m and its CRC when calculating traffic reduction. As will be explained below, is often advantageous to remove CRC entries from groups that are not queried that often by the client. Each calculation is for a certain period of time and will predict a pre-fetch cache. It can be advantageous to remove those CRCs which are not queried that often by comparing the new pre-fetch cache and old pre-fetch cache. It is preferable to calculate a pre-fetch cache for each group as a unit, not for single user. The following formula may be used to calculate the traffic reduction for particular values of m:

$$\left[\sum_{i=1}^{m}\int_{t1}^{t2} f_i(t)\,dt\right] \times CRC_{avg\ query\ size} - \left[\int_{t1}^{t2} g(t)\,dt\right] \times \left(CRC_{avg\ query\ size} + Deploy_{size}\right) = Traffic\ Reduction \quad (1)$$

Fig. 11 graphically illustrates a timeline showing total CRC counts and the relative positions of the time parameters used in the above formula. In this graph time is on the x-axis and total CRC count is on the y-axis. We define: t0 as being the starting point of the data analysis; t1 as being the starting point of deploying the pre-fetch cache; t2 as being the cache

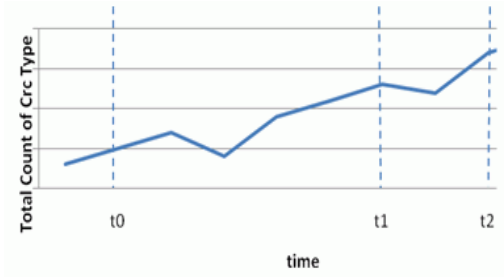


FIGURE 11. Timeline of CRC counts.

TTL (time-to-live) period of seven days; and the time period of t0 – t1 being the analysis of the current data.

Further variables in the formula are defined as follows:

$$Deploy\_size = m \times [\![CRC]\!]\_(avg\ cache\ size) \quad (2)$$

$$\sum_{i=1}^{m} \int_{t1}^{t2} f_i(t)\, dt = Total\ CRC\ count\ between\ t1\ and\ t2\ for\ indidual\ CRC\ type \quad (3)$$

where $f_i(t)$ is a prediction function for a given CRC type i.

A special condition may apply when this prediction function results in a negative value, i.e., when $f_i(t) < 0$. What this means is that at any given point of time, if the area under a curve is a negative value, this might due to probability of error. This can be treated as a special case from a data analysis standpoint.

$$\int_{t1}^{t2} g(t)\, dt = Total\ Clients\ need\ to\ deploy\ between\ t1\ and\ t2 \quad (4)$$

where $g(t)$ is the prediction of the total number of clients (user computers) who will receive the pre-fetch cache.

One way to explain the formula is to understand that the quantity of the first part on left hand side represents the forecast traffic (assuming that the pre-fetch cache technique is not implemented), and the quantity of the second part on the left hand side represents the deployment traffic (when the pre-fetch cache is deployed to clients). To determine that it is worthwhile to deploy a group of the pre-fetch cache to a client, the Traffic Reduction result value from the CRC forecast minus CRC for deployment must not be a negative value. If there is a positive value then a traffic reduction flag is associated with this group of CRCs. The range from t1 to t2 is seven days in this embodiment.

As mentioned above, if the traffic volume associated with downloading a particular group is greater than the traffic volume associated with performing normal queries, then it is desirable to remove a CRC entry or entries from the group and perform calculations again in order to obtain the maximum achievable traffic reduction. In other words, entries may be removed from a group until the calculation of subtracting the download pre-cache volume from the user query volume yields the greatest value. In one embodiment, this iteration is performed by first removing those CRC entries from a group which have been sent the fewest times from a particular product. If the traffic volume associated with downloading a particular group is always greater than the traffic volume associated with performing normal queries (even after removing any number of CRC entries) then the group is not flagged with a traffic reduction flag and the group will not be downloaded as a pre-fetch cache.

E. CHECK WITH VIRUS PATTERN

In many instances, a particular virus pattern (containing any number of CRCs identifying known viruses) for a particular product may be updated as frequently as once per hour. It is possible that any of the CRCs in one of the groups now matches with a known CRC in the updated virus pattern. Because the virus pattern is typically applied on a client computer before the pre-fetch cache is checked, a known virus will be caught by the virus pattern before the pre-fetch cache is checked. Therefore, in this situation, there is no need for that particular CRC that matches to also be present within one of the groups. These CRCs that match with any CRC of an updated virus pattern may then be deleted in step 5 from the groups in order to save on the bandwidth required to send these groups over the Internet to the user computers. s step is useful but optional.

F. FORM CUSTOMIZED PRE-FETCH CACHE

Next, in step 6, after any matching CRCs have been deleted from the groups in step 5 a custom pre-fetch cache containing any number of the groups is sent to the user computers in the field. In one embodiment, all groups having the key attribute of ''region'' which is the same will be downloaded to all user computers in that particular region. For example, if fourteen of the groups have a key attribute of ''region'' which is North America, then all those groups will be sent to all user computers in the region of North America. Or, one may download based upon product and not upon region. In order to send the pre-fetch cache to user computers in field, it is sufficient to send just the CRCs and the corresponding result. If the result is ''virus,'' the corresponding CRC will be added into the virus pattern and there is no need to handle it any more.

Once each client computer has received its pre-fetch cache and is now processing suspect files and sending out new queries to the backend servers over the Internet, it is possible to begin the process again of obtaining an updated access log from that point in time and reiterating the steps. The client performs the steps as shown in Fig.12:

IV. IMPLEMENTATION

First of all, duo to resource limitation, all access logs are too huge and too hard to analysis by our limited machines, the prototype only bases focus ''one'' antivirus product and collect only one day historical access logs to analysis. Some server side products are not easy to do clustering. Because these kinds of access log queries may come from single one server. It's hard to identify how many users behind the server.

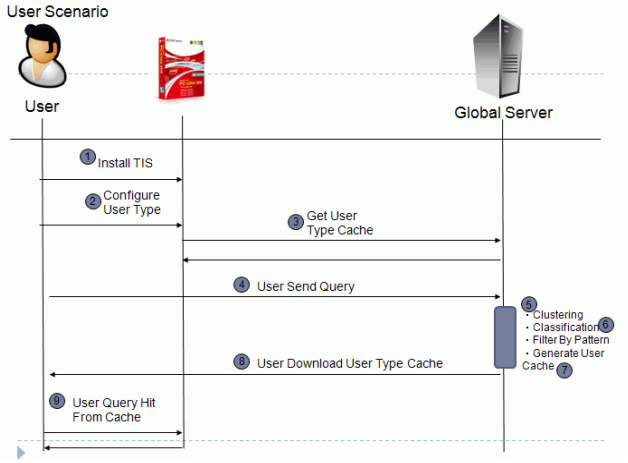


FIGURE 12. User Scenario - download cache.

Second, here choose one product as our log analysis RAW data. Duo to its personal antivirus version, the access logs may contain some different categories and group data that useful for the Two Step clustering.

Finally, the prototype tries to apply our six steps analysis to form the cache. After pre-fetch cache is generated, check the cache effectiveness by parsing the access logs we have.

A. IMPLEMENTATION HYPOTHESIS

This section we introduce the basic of calculation.

Suppose Fig. 13 shows all historical traffic happened between time t0 to time t2. X-axis stands for the time period; Y-axis stands for traffic volume. This traffic volume aggregated those real access logs collected from those cloud backend servers.

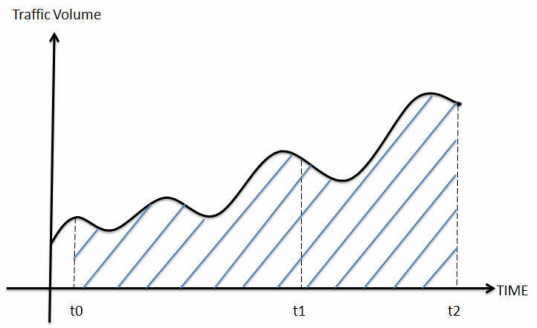


FIGURE 13. All historical traffic volume (t0-t2).

First, we collect all access logs from t0 to t1, then utilize data mining technique [23]–[25], [30], [40] and clustering concept, by gathering the current feedback data we have from our SPN, we are able to form these data in groups with similarity, and by deploying these data to client side on t1, compare the original traffic volume from t1 to t2, and then we can do the performance evaluation.

Then, the new traffic volume after applied pre-fetch cache becomes the Fig.14 From t1 to t2, it's clear the traffic volume is fewer than before (orange color).

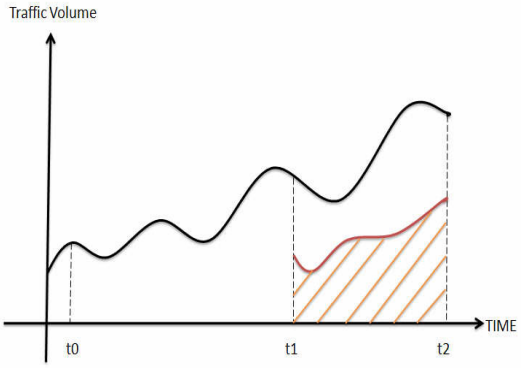


FIGURE 14. User download pre-fetch cache traffic volume (t1-t2).

The following calculation can prove that if clients can download the pre-fetch cache in t1, then they will not send out too many queries to the company's backend servers from t1 to t2. Following that, we can calculate how many traffic we saved based on the real case. Since we already have all real data from t1 to t2, we can compare the data and calculate the real hit cache without interfering real customer machines.

B. DETAILED PROCESS

1) RAW ACCESS LOG PARSING

Raw data comes from the server's background access log RAW data, Fig. 15 lists those selected data we parsed from access log and prepare to do further process.

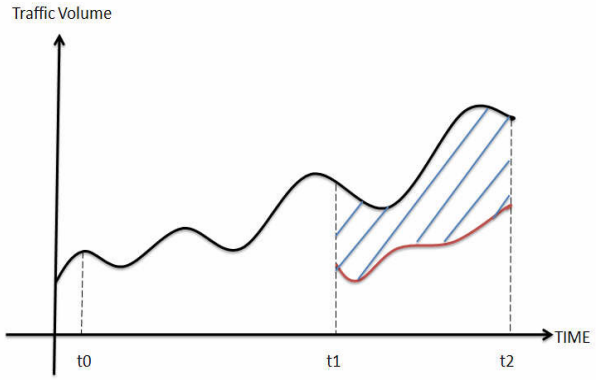


FIGURE 15. Saved traffic volume after applied Pre-fetch. Cache (t1-t2).

2) USE SPSS – TWOSTEP PROCESS TO FORM GROUP

Here try to handle real case, input raw data (useful query logs) to SPSS, and then try to form group.

For use with single product access logs got from backend servers in the cloud, after process raw data to SPSS TwoStep process, we got 99 groups in single product.

In the prototype, we calculate these 99 groups and try to identify which groups are suitable for real cache implementation.

TABLE 1. Summary of CRC statics by group.

Group Name	CRC Count	Client Count	CRC Invidual Count	Client Invidual Client	CRC Requery Rate	Client Requery Rate
GROUP_83	7,584	84	121,673	121,673	16.0433808	1448.488095
GROUP_25	238,227	710	417,524	417,524	1.752630894	588.0619718
GROUP_19	2,251,332	20,453	3,312,608	3,312,608	1.471399154	161.9619616
GROUP_10	306,033	1,749	434,714	434,714	1.420480798	248.5500286
GROUP_58	43,466	21	61,551	61,551	1.416072332	2931
GROUP_73	9,551	99	12,952	12,952	1.356088368	130.8282828
GROUP_35	86,393	608	116,685	116,685	1.350630259	191.9161184
GROUP_86	328	3	442	442	1.347560976	147.3333333
GROUP_27	832,360	6,326	1,119,801	1,119,801	1.345332548	177.0156497
GROUP_46	7,418	112	9,872	9,872	1.330816932	88.14285714
GROUP_12	38,229	110	50,736	50,736	1.327160009	461.2363636
GROUP_44	70,568	547	90,185	90,185	1.27798719	164.8720293
GROUP_15	34,608	265	43,521	43,521	1.257541609	164.2301887
GROUP_36	136,243	1,497	170,292	170,292	1.249913757	113.755511

TABLE 2. Data for calculate traffic reduction.

	Vcrc (Slope)	Avg CRC Query Size	Vclinet (Slope)	Avg Cache Size	CRC Count
Group_9	0.00002068777659382360	222.6216832	-0.00000017939195900984	52.35487097	401,129
Group_58	-0.00000138351190472022	224.5362	-0.00000000269687319214	52.35487097	43,466
Group_27	0.00000668678393495444	222.4569345	-0.00000067137399418215	52.35487097	832,360
Group_19	0.00013055301342307500	222.3247004	-0.00000191454602229275	52.35487097	2,251,332
Group_83	0.00000082986869309594	222.1965981	-0.00000000619500701915	52.35487097	7,584

3) USE CRC STATICS BY GROUP

Use SPSS – TwoStep process to Form Group, then use the group the calculate each group has how many CRC and clients and the CRC hit rate…etc.

The converted test data that CRC statics in different groups. Then, summarize the result into Table 1

Table 1 shows the summary data about CRC statics by Group. For example, we can found the following information in Group 19:

Group 19

There are 20,453 clients belong to Group 19.

All CRC Counts 2,251,332

Individual CRC Count is 3,312,608, which means that identical CRC in group 19.

CRC Re-query Rate is 1.47

Client Re-query Rate is 161.96

Here we compare the CRC Re-query Rate in different groups, then try to choose those groups with maximum rate, and then use their CRC data to form the pre-fetch cache.

Following that, we choose 5 groups. They are group 9, group 53, group 27, group 19, group 83 in Table 2 from Fig. 16. Then we do real calculation.

The formula that we used here:

Total time : Suppose " t1" (describe in Section) is 7 hours, which equals to $60 \times 24 \times 7$ seconds $= 10080$ seconds

Total Volume Size $=$ (Vcrc (Slope) $*$ t1) $*$ (t1 / 2) $*$ Avg CRC Query Size

Total Deploy Cache Size $=$ (Vclient(slope)) $*$ (t1 / 2) $*$ (Avg Cache Size $+$ Avg CRC Query Size) $*$ (CRC Count)

Then we got the data list in Table 4-3

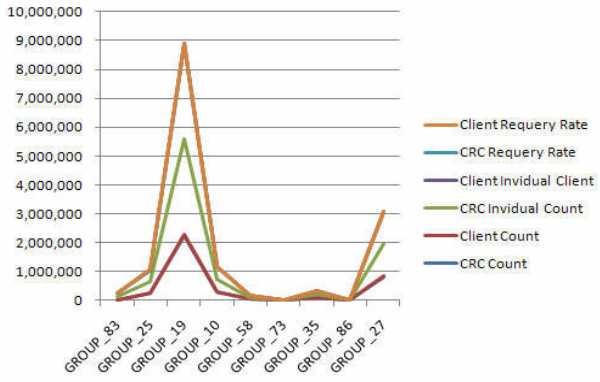


FIGURE 16. Summary of CRC statics by group.

C. PERFORMANCE EVALUATION

After process through detail steps in table III, we can calculate how many network traffic can be reduce through current increasing traffic

The following items are total real network traffic that client should use:

TotalBE : Total clients that we formed in Group

Query Size: Each normal query has to send to backend server

CrcR: CRC Hit Rate

TotalBE $= 121673$

QuerySize $= 1.7$

CrcR $= 0.1604$

The following items are total network traffic when the prototype has to deploy the cache first.

TABLE 3. Traffic reduction calculate result.

	Total Volume Size	Total Deploy Cache Size	Result
Group_9	233976.5582	-1005249272	1005483249
Group_58	-15781.93817	-1648960.383	1633178.445
Group_27	75570.85003	-7801932170	7802007740
Group_19	1474570.988	-60148220677	60149695248
Group_83	9367.805285	-655322.3313	664690.1365

DeploySize : Size should be taken by each cache item
Cache Count : total cache the prototype used
DeployUser : Real users that adopt the pre-fetch cache
DeploySize = 0.3
CacheCount = 1216
CacheSize = DeploySize∗CacheCount
DeployUser = 84
TrafficLeft : Total network traffic that clients waste originally
TrafficRight : Total network traffic that deploy pre-fetch cache to clients
TrafficLeft = TotalBE∗CrcR∗QuerySize = 33177.8
TrafficRight = DeployUser∗(QuerySize + CacheSize) = 30786.0
SavedTraffic = TrafficLeft - TrafficRight = 2391.79
SavedRate = (TrafficLeft - TrafficRight)/TrafficLeft
= 0.0720902
= 7.20902%
Here the prototype proved that the network traffic can be really reduced by the solution.

V. CONCLUSION

In most of the SPN products, local cache mechanism is implemented for the purpose of reducing the network traffics. However, the local cache is treated as a single group, by all means that, there is no group interaction concept applied, and as the result, it is not efficient way to use these data from these singe group to build or forming new cache.

After these six steps to generate pre-fetch cache for clients, the paper utilize data mining technique and clustering concept, by gathering the current feedback data we have from SPN backend server, the solution can form these data in groups with similarity, and by deploying these data to client side, to achieve the reduction of traffic usage.

A. BENEFIT OF SEND THE PRE-FETCH CACHE TO CLIENT

For the Security solution provider, the cloud server to serve SPN service is limited, the charge by CDN company also based on the network traffic. Bases on the solution, network traffic can reduce about10%, so the cost for CDN can be cut instantly.

B. FUTURE WORK

1) APPLY OUR SOLUTION MODEL TO WRS AND OTHER COMPANY SERVICES

In this paper, we only use FRS as an example; however, our data should also be able to fit in to the case of WRS. This is because if data that can be represented as a polynomial function, then we will be able to transform the data into numbers, and calculate if it's worthwhile for deployment to achieve the reduce traffic purpose.

2) PRODUCT DEFAULT CUSTOMIZED CACHE

Suggest that we can generate different cache files by different groups. When users start to install the product, use some questions to identify the detailed user profile, then system can pre-install customized cache into products for the user.

REFERENCES

[1] A. Vakali and G. Pallis, "Content delivery networks: Status and trends," *IEEE Internet Comput.*, vol. 7, no. 6, pp. 68–74, Nov./Dec. 2003.
[2] R. Buyya, A.-M. K. Pathan, J. Broberg, and Z. Tari, "A case for peering of content delivery networks," *IEEE Distrib. Syst. Online*, vol. 7, no. 10, p. 3, Oct. 2006.
[3] Akamai Technologies. (2018). *The world's Largest and Most Trusted Cloud Delivery Platform*. [Online]. Available: https://www.akamai.com
[4] P.-J. Maenhaut, H. Moens, M. Verheye, P. Verhoeve, S. Walraven, E. Truyen, W. Joosen, V. Ongenae, and F. D. Turck, "Migrating medical communications software to a multi-tenant cloud environment," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage.*, May 2013, pp. 900–903.
[5] T. Dillon, C. Wu, and E. Chang, "Cloud computing: Issues and challenges," in *Proc. 24th IEEE Int. Conf. Adv. Inf. Netw. Appl.*, Apr. 2010, pp. 27–33.
[6] J. Yang and Z. Chen, "Cloud computing research and security issues," in *Proc. Int. Conf. Comput. Intell. Softw. Eng.*, Dec. 2010, pp. 1–3.
[7] D. Petcu, G. Macariu, S. Panica, and C. Crăciun, "Portable Cloud applications—From theory to practice," *Future Gener. Comput. Syst.*, vol. 29, no. 6, pp. 1417–1430, 2013.
[8] K. Habak, M. Ammar, K. A. Harras, and E. Zegura, "Femto Clouds: Leveraging mobile devices to provide cloud service at the edge," in *Proc. IEEE 8th Int. Conf. Cloud Comput.*, Jun./Jul. 2015, pp. 9–16.
[9] M. D. Assunção R. N. Calheiros, S. Bianchi, M. A. S. Netto, and R. Buyya, "Big Data computing and clouds: Trends and future directions," *J. Parallel Distrib. Comput.*, vols. 79–80 pp. 3–15, May 2015.
[10] O. Aciiçmez and W. Schindler, "A vulnerability in RSA implementations due to instruction cache analysis and its demonstration on OpenSSL," in *Proc. Cryptographers' Track RSA Conf.*, 2008, pp. 256–273.
[11] G. Doychev, D. Feld, B. Köpf, L. Mauborgne, and J. Reineke, "Cacheaudit: A tool for the static analysis of cache side channels," in *Proc. USENIX Conf. Secur.*, 2013, pp. 431–446.
[12] T. Kim, M. Peinado, and G. Mainar-Ruiz, "*STEALTHMEM*: System-level protection against cache-based side channel attacks in the Cloud," in *Proc. 21st USENIX Conf. Security Symp.*, 2013, pp. 189–204.

[13] A. Boroumand, S. Ghose, M. Patel, H. Hassan, B. Lucia, K. Hsieh, K. T. Malladi, H. Zheng, and O. Mutlu, "LazyPIM: An efficient cache coherence mechanism for processing-in-memory," *IEEE Comput. Archit. Lett.*, vol. 16, no. 1, pp. 46–50, Jan./Jun. 2017.

[14] Wikipedia Contributors. (2018). *CRC*. [Online]. Available: https://zh.wikipedia.org/wiki/CRC

[15] M. Ahmadi and S. Wong, "A cache architecture for counting bloom filters," in *Proc. 15th IEEE Int. Conf. Netw.*, Nov. 2007, pp. 218–223.

[16] B. Chazelle, J. Kilian, R. Rubinfeld, and A. Tal, "The Bloomier filter: An efficient data structure for static support lookup tables," in *Proc. 15th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2004, pp. 30–39.

[17] P. Rodriguez, C. Spanner, and E. W. Biersack, "Analysis of Web caching architectures: Hierarchical and distributed caching," *IEEE/ACM Trans. Netw.*, vol. 9, no. 4, pp. 404–418, Aug. 2001.

[18] J. Yin, Z.-F. Tan, J.-T. Ren, and Y.-Q. Chen, "An efficient clustering algorithm for mixed type attributes in large dataset," in *Proc. Int. Conf. Mach. Learn. Cybern.*, Vol. 3, Aug. 2005, pp. 1611–1614.

[19] A. Ahmad and L. Dey, "A κ-mean clustering algorithm for mixed numeric and categorical data," *Data Knowl. Eng.*, vol. 63, no. 2, pp. 503–527, Nov. 2007.

[20] Z. He, X. Xu, and S. Deng, "Scalable algorithms for clustering large datasets with mixed type attributes," *Int. J. Intell. Syst.*, vol. 20, no. 10, pp. 1077–1089, 2005.

[21] C. Ittrich, D. Krause, and W.-D. Richter, "Probabilities and large quantiles of noncentral generalized chi-square distributions," *J. Theor. Appl. Statist.*, vol. 34, no. 1, pp. 53–101, 2000.

[22] I. Campbell, "Chi-squared and Fisher–Irwin tests of two-by-two tables with small sample recommendations," *Statist. Med.*, vol. 26, no. 19, pp. 3661–3675, Aug. 2007.

[23] L. Rokach and O. Maimon, *Data Mining with Decision Trees: Theory and Applications*. Singapore: World Scientific, 2007.

[24] H. Jiawei and K. Micheline, "Data mining: Concepts and techniques," *Data Mining Concepts Models Methods Algorithms*, vol. 5, no. 4, p. 1–18, 2006.

[25] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg, "Top 10 algorithms in data mining," *Knowl. Inf. Syst.*, vol. 14, no. 1, pp. 1–37, Jan. 2008.

[26] Wikipedia contributors. (2018). *Hierarchical Clustering*. https://en.wikipedia.org/w/index.php?title=Hierarchical_clustering&oldid=328921893.%20Accessed%20December%203

[27] S. Dasgupta, "A cost function for similarity-based hierarchical clustering," in *Proc. 48th Annu. ACM Symp. Theory Comput.*, 2016, pp. 118–127.

[28] P. Awasthi, A. S. Bandeira, M. Charikar, R. Krishnaswamy, S. Villar, and R. Ward, "Relax, no need to round: Integrality of clustering formulations," in *Proc. Conf. Innov. Theor. Comput. Sci.*, 2015, pp. 191–200.

[29] Wikipedia Contributors. (2018). *Cluster Analysis*. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Cluster_analysis&oldid=327620852,%20November%2024

[30] P. Saxena, V. Singh, S. Lehri, P. Saxena, V. Singh, and S. Lehri, "Evolving efficient clustering patterns in liver patient data through data mining techniques," *Int. J. Comput. Appl.*, vol. 66, no. 16, pp. 23–28, 2013.

[31] D. N. Allen and G. Goldstein, *Cluster Analysis in Neuropsychological Research*. New York, NY, USA: Springer, 2013.

[32] P. K. Kimes, Y. Liu, D. N. Hayes, and J. S. Marron, "Statistical significance for hierarchical clustering," *Biometrics*, vol. 73, no. 3, pp. 811–821, 2017.

[33] M. Valk and G. B. Cybis, "U-statistical inference for hierarchical clustering," May 2018, *arXiv:1805.12179*. [Online]. Available: https://arxiv.org/abs/1805.12179

[34] A. Krishnamurthy, S. Balakrishnan, M. Xu, and A. Singh, "Efficient active algorithms for hierarchical clustering," Jun. 2012, *Computer Science*. [Online]. Available: https://arxiv.org/abs/1206.4672

[35] C. Cooper, D. Franklin, M. Ros, F. Safaei, and M. Abolhasan, "A comparative survey of vanet clustering techniques," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 1, pp. 657–681, 1st Quart., 2017.

[36] W. Zhu, W. Guo, Z. Yu, and H. Xiong, "Multitask allocation to heterogeneous participants in mobile crowd sensing," *Wireless Commun. Mobile Comput.*, vol. 2018, Jun. 2018, Art. no. 7218061.

[37] Y. Mo, L. Xing, Y.-K. Lin, and W. Guo, "Efficient analysis of repairable computing systems subject to scheduled checkpointing," *IEEE Trans. Dependable Secure Comput.*, to be published. doi: 10.1109/TDSC.2018.2869393.

[38] Y. Yang, X. Liu, X. Zheng, C. Rong, and W. Guo, "Efficient traceable authorization search system for secure cloud storage," *IEEE Transactions on Cloud Computing*, to be published. doi: 10.1109/TCC.2018.2820714.

[39] W. Guo, N. Xiong, A. V. Vasilakos, G. Chen, and H. Cheng, "Multi-source temporal data aggregation in wireless sensor networks," *Wireless Pers. Commun.*, vol. 56, no. 3, pp. 359–370, Feb. 2011.

[40] W. Guo and G. Chen, "Human action recognition via multi-task learning base on spatial–temporal feature," *Inf. Sci.*, vol. 320, pp. 418–428, Nov. 2015.

[41] K. Guo, W. Guo, Y. Chen, Q. Qiu, and Q. Zhang, "Community discovery by propagating local and global information based on the MapReduce model," *Inf. Sci.*, vol. 323, pp. 73–93, Dec. 2015.

[42] D. Rosário, Z. Zhao, A. Santos, T. Braun, and E. Cerqueira, "A beaconless opportunistic routing based on a cross-layer approach for efficient video dissemination in mobile multimedia IoT applications," *Comput. Commun.*, vol. 45, no. 1, pp. 21–31, Jun. 2014.

[43] G. Liu, X. Huang, W. Guo, Y. Niu, and G. Chen, "Multilayer obstacle-avoiding X-architecture Steiner minimal tree construction based on particle swarm optimization," *IEEE Trans. Cybern.*, vol. 45, no. 5, pp. 1003–1016, May 2015.

[44] F. Luo, W. Guo, Y. Yu, and G. Chen, "A multi-label classification algorithm based on kernel extreme learning machine," *Neurocomputing*, vol. 260, pp. 313–320, Oct. 2016.

[45] T.-Y. Wu, C.-M. Chen, K.-H. Wang, C. Meng, and E. K. Wang, "A provably secure certificateless public key encryption with keyword search," *J. Chin. Inst. Eng.*, vol. 42, no. 1, pp. 20–28, 2019.

[46] T.-Y. Wu, C.-M. Chen, K.-H. Wang, and J. M.-T. Wu, "Security analysis and enhancement of a certificateless searchable public key encryption scheme for IIoT environments," *IEEE Access*, vol. 7, pp. 49232–49239, 2019.

[47] C.-M. Chen, B. Xiang, Y. Liu, and K.-H. Wang, "A secure authentication protocol for Internet of vehicles," *IEEE Access*, vol. 7, pp. 12047–12057, 2019.

[48] L. Ni, F. Tian, Q. Ni, Y. Yan, J. Zhang, "An anonymous entropy-based location privacy protection scheme in mobile social networks," *EURASIP J. Wireless Commun. Netw.*, vol. 2019, no. 1, p. 93, 2019.

[49] H. Xiong, Y. Zhao, L. Peng, H. Zhang, and K.-H. Yeh, "Partially policy-hidden attribute-based broadcast encryption with secure delegation in edge computing," *Future Gener. Comput. Syst.*, vol. 97, pp. 453–461, Aug. 2019.

[50] H. Xiong and Z. Qin, "Revocable and scalable certificateless remote authentication protocol with anonymity for wireless body area networks," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 7, pp. 1442–1455, Jul. 2015.

[51] J.-S. Pan, L. Kong, T.-W. Sung, P.-W. Tsai, and V. Snasel, "Alpha-fraction first strategy for hierarchical wireless sensor networks," *J. Internet Technol.*, vol. 19, no. 6, pp. 1717–1726, 2018.

[52] J. M.-T. Wu, M.-H. Tsai, Y.-Z. Huang, S. K. H. Islam, M. M. Hassan, A. Alelaiwie, and G. Fortinof, "Applying an ensemble convolutional neural network with Savitzky–Golay filter to construct a phonocardiogram prediction model," *Appl. Soft Comput.*, vol. 78, pp. 29–40, May 2019.

[53] J.-S. Pan, C.-Y. Lee, A. Sghaier, M. Zeghid, and J. Xie, "Novel systolization of subquadratic space complexity multipliers based on Toeplitz matrix–Vector product approach," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 7, pp. 1614–1622, Jul. 2019.

[54] C.-M. Chen, K.-H. Wang, K.-H. Yeh, B. Xiang, T.-Y. Wu, "Attacks and solutions on a three-party password-based authenticated key exchange protocol for wireless communications," *J. Ambient Intell. Hum. Comput.*, vol. 10, no. 8, pp. 3133–3142, Aug. 2019.

[55] W. Gan, J. C.-W. Lin, P. Fournier-Viger, H.-C. Chao, and P. S. Yu, "HUOPM: High-utility occupancy pattern mining," *IEEE Trans. Cybern.*, to be published. doi: 10.1109/TCYB.2019.2896267.

[56] J. C.-W. Lin, Y. Zhang, B. Zhang, P. Fournier-Viger, and Y. Djenouri, "Hiding sensitive itemsets with multiple objective optimization," *Soft Comput.*, to be published. doi: 10.1007/s00500-019-03829-3.

[57] J. C.-W. Lin, P. Fournier-Viger, L. Wu, W. Gan, Y. Djenouri, and J. Zhang, "PPSF: An open-source privacy-preserving and security mining framework," in *Proc. IEEE Int. Conf. Data Mining Workshops (ICDMW)*, Singapore, Nov. 2018, pp. 1459–1463.

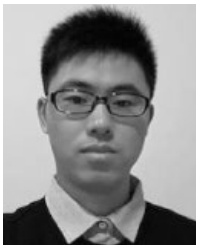
[58] J. C.-W. Lin, L. Yang, P. Fournier-Viger, T.-P. Hong, "Mining of skyline patterns by considering both frequent and utility constraints," *Eng. Appl. Artif. Intell.*, vol. 77, pp. 229–238, Jan. 2019.

LANLAN KANG received the M.S. degree in computer applications technology from the School of Information and Engineering, Jiangxi University of Science and Technology, Ganzhou, China, in 2009, and the Ph.D. degree in computer software and theory from the Computer School, Wuhan University, Wuhan, China, in 2017. In 2002, she joined the faculty of the Jiangxi University of Science and Technology, where she is currently a Lecturer. She has published about 20 relevant research articles. Her research interests include intelligent computing, machine learning, time series prediction, and the Internet of Things.

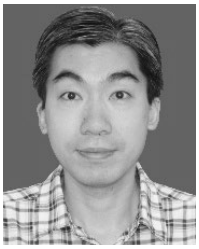
RUEY-SHUN CHEN received the M.S. and Ph.D. degrees from the Department of Computer Science and Information Engineering, National Chiao Tung University, Taiwan. He is currently a Professor with the Department of Computer Engineering, Dongguan Polytechnic, Dongguan, China. His research interests include radio-frequency identification (RFID), data mining, genetic algorithms, performance evaluation, business networks, distributed systems, the IoT and security.

YEH-CHENG CHEN is currently pursuing the Ph.D. degree with the Department of Computer Science, University of California, Davis, CA, USA. His research interests include radio-frequency identification (RFID), data mining, social networks, information systems, wireless network artificial intelligence, the IoT and security.

CHUNG-CHEI WANG received the M.S. degree from National Chiao Tung University, Taiwan, where he is currently a Manager. His research interest includes security RFID Networks.



XINGGUAN LI received the B.Sc. and Ph.D. degrees in applied mathematics from Fuzhou University, Fuzhou, China, in 2013, and 2018, respectively. He is currently a Lecturer with the School of Mathematics and Statistics, Minnan Normal University, Zhangzhou, China. He has authored over 10 papers in refereed journals and conferences, including the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS, the *ACM Transactions on Design Automation of Electronic Systems*, *Integretion*, the VLSI Journal, the Journal *Basic & Clinical Pharmacology & Toxicology*, *IEEE/ACM International Conference on Computer Aided Design*, *IEEE/ACM Asia and South Pacific Design Automation Conference*, and *IEEE Computer Society Annual Symposium on VLSI*. His current research interests include VLSI physical design and design for manufacturability, and optimization theory and algorithms. He was a recipient of the First Place in the ICCAD 2017 Contest and the First Place in the ICCAD 2018 Contest. He received Excellent Doctor Degree Dissertation Award in Fujian and Graduate National Scholarship.



TSU-YANG WU received the Ph.D. degree from the Department of Mathematics, National Changhua University of Education, Taiwan, in 2010. He was an Assistant Professor with the Innovative Information Industry Research Center, Shenzhen Graduate School, Harbin Institute of Technology. He is currently an Associate Professor with the College of Computer Science and Engineering, Shandong University of Science and Technology, China. He serves as an Executive Editor of *Journal of Network Intelligence* and as an Associate Editor of *Data Science and Pattern Recognition*. His research interests include network security and data mining.