**AUSTRALIAN JOURNAL OF BASIC AND APPLIED SCIENCES**

ISSN:1991-8178 EISSN: 2309-8414
Journal home page: www.ajbasweb.com

# Anti Virus Scanner with Pipelined Virus Database

[1]D. Jennifer, [2]K. Nivethitha, [3]K. Sangeetha and [4]K. Kiruthika

[1]*Computer Science and Engineering, Panimalar Engineering College, Tamil Nadu, India.*
[2]*Computer Science and Engineering, Panimalar Engineering College, Tamil Nadu, India.*
[3]*Computer Science and Engineering, Panimalar Engineering College, Tamil Nadu, India.*
[4]*Computer Science and Engineering, Panimalar Engineering College, Tamil Nadu, India.*

**Address For Correspondence:**
D. Jennifer, Computer Science and Engineering, Panimalar Engineering College, Tamil Nadu, India..
E-mail: pecjennifer@gmail.com,

**ARTICLE INFO**

***Article history:***
*Received 10 December 2015*
*Accepted 28 January 2016*
*Available online 10 February 2016*

***Keywords:***
*Virus database, Scan engine, Heuristic analysis*

**ABSTRACT**

The main aim of this work is to build a virus scanner that consumes minimal main memory space for the storage of the virus database while scanner is in operation. All commercial antivirus products have three major components :(1) scan engine, (2) VDBs, (3) updater. VDB contains virus signatures for all the viruses. Building a VDB to cope with almost one million viral patterns is not a painless task and while the scanner is in progress, massive memory is needed. To overcome this, we are introducing a proposal to build a virus scanner that consumes minimal main memory space. In this work, the viral patterns will be grouped into different sets according to their nature and allowed to enter into main memory whenever needed, i.e. instead of loading all patterns, only a few patterns are loaded at a time, and the scan engine scans the file searching for a particular set of patterns. Once the scan is completed for the listed patterns, then another set of patterns will be loaded and the scan is repeated, which leads to significant reduction in the memory usage. To accomplish this, virus scan engine should be customized to be adaptable with the modified VDB structure.

## INTRODUCTION

The antivirus software is one of the security solutions widely used in most of the computing devices like PC, laptop, PDA, smart phone, etc. The factors that affect the performance of an antivirus are scan speed, memory consumption and up-to-date virus database (VDB). Among these the first two majorly depend on the size of the VDB. The smaller the VDB quicker is the scan completion, and a larger VDB incurs more scan time. We cannot reduce the VDB size, since VDB should have virus signatures (Qingguo Wang1,Peilin Jia Zhongming Zhao1, 2013) of all the known viruses. Whenever a new virus is discovered, it is not possible for an existing virus scanner to detect the same. Therefore, a new signature should be created for each newly discovered virus and the same should be updated in VDB by the vendor once it is successfully tested. These updates increase the detection capability of a scan engine.There are more than one million computer viruses that are known to exist till date (Yunxin Chen, 2013). Building antivirus software for protecting against all of them is not an easy task. As VDB size increases, the complexity of scan engine increases.

### *Overview Of Antivirus Software:*

Fig 1 depicts the overview of the conventional antivirus software.

In conventional antivirus software when the scanning process is started, the first module/component to execute the scanning process is the scan engine which has the ability to scan the file and confirm the presence and absence of the virus with the help of the viral patterns or signatures incorporated in the VDB. Both the scan engine and the VDB should be loaded in memory, and they should reside there always until the scan engine



**To Cite This Article:** D. Jennifer, K. Nivethitha, K. Sangeetha and K. Kiruthika., Anti Virus Scanner with Pipelined Virus Database. ***Aust. J. Basic & Appl. Sci.,*** *10(1): 238-242, 2016*

completes the scanning process of all files in the system. When a file is identified as infected with virus, they should be either deleted or should be isolated in quarantine location (like isolating a person with contagious disease) which is also stored in secondary storage in the file system. Isolation and quarantine are public health practices used to stop or limit the spread of disease (Chieh-Jen Cheng, 2012). Isolation is used to separate ill persons who have a communicable disease from those who are healthy. Quarantine is used to separate and restrict the movement of healthy persons who may have been exposed to a communicable disease to see if they become ill. The files put in the quarantine should be rendered inactive. For making them inactive, the files should be compressed and encrypted prior to being sent to quarantine (inactivation is needed to prevent them from infecting other healthy persons).

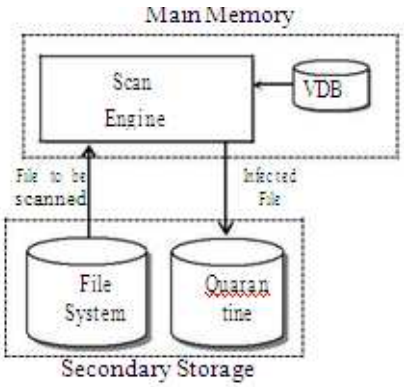


**Fig. 1:** Conventional antivirus architecture.

***Potential Problems:***

There are several types of viruses that exist (Piyachon P. and Y. Luo, 2006). The detection of all of them needs different kinds of techniques. As a result, complexity of the scan engine and its size keep on increasing evidently day by day, which affects the system's performance too. We all have an experience with the PC antivirus which slows down system performance ("running antivirus on a personal computer is like having the bomb squad inspect a suspicious package inside the house right next to you") (http://www.tomshardware.com/reviews). This slow down in performance is observed even in the modern state of the art PC with higher configuration. Today's PCs are much more powerful than they were a few years ago, so perhaps the notion that an antivirus application will still have a debilitating effect on performance is unfounded (http://viralpatel.net/taj/tutorial).


**Fig. 2:** Scanner consuming 200 MB memory at the beginning.

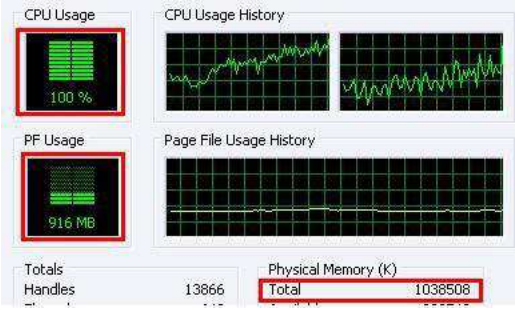


**Fig. 3:** Scanner consuming heavy CPU and main memory.

The main problem encountered here is that when scanning is in progress, we cannot run any other

application in the system. In fact, sometimes, we cannot even move the cursor. Antivirus is an add-onal utility for protecting the documents and all other applications. But the antivirus requires more resources (CPU, memory, etc.) so other processes would not run or have to wait for AV to release the resources they consumed. Fig 2 and 3 depict this problem.

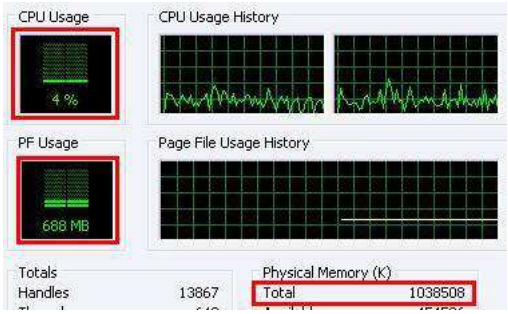


**Fig. 4:** CPU and main memory usage when scanner is not in execution.

Fig4 shows CPU and main memory usage when scanner is not in progress. Next section describes the solution to this problem.

***The Proposed Av Model:***

To overcome the problems stated in section 3, we have designed an antivirus software which consumes much less system resource and gives better performance. The main cause for the slowdown in performance is the loading of bulk VDB completely into the main memory. The loaded VDB will reside in the main memory throughout the scanning process

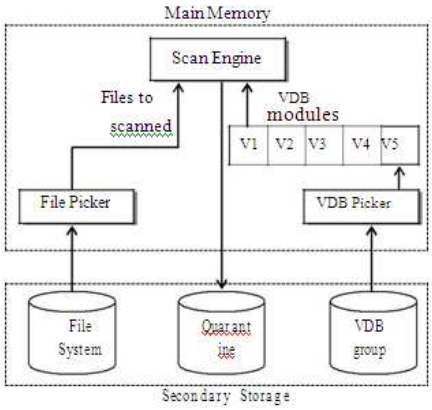


**Fig. 5:** Proposed AV model.

Fig 5 gives the quick overview of our new proposed antivirus model. In this model, the major inclusion is the pipelined VDB database. The viral patterns or signatures should be grouped into different sets according to their type/nature. Instead of storing all VDBs in a single construct, we store them in different groups. Each group will be allocated to one VDB. Likewise, various VDBs should be generated, and all such VDBs should be stored in application path.

Normally, when a file needs to be scanned, the scanning should be performed against all patterns. But at any point of time, we need only one pattern. Therefore, viral patterns can be loaded whenever it is needed similar to on-demand service. But in this case, after scanning the file against the first pattern, we should feed the scan engine with the next pattern. Loading the next pattern involves some time delay, which slows down the scan process. To avoid the time lag due to the scan engine waiting for pattern to be loaded, we have introduced a concept, the so-called pipelined VDB.

In the proposed model, the VDB groups will be loaded in sequence one after another. At the beginning, five VDB groups should be loaded in pipeline. Once the scan finishes with first VDB, it should be removed and the VDB following the previous one should be advanced up. New VDB should be shifted to the last position of the pipeline.

Fig 6 shows how the VDBs can be loaded into main memory in a pipelined manner. Here the key is loading the VDB into main memory and removing them from the main memory when they are no longer required. Proper allocation of memory and release of memory allocated to VDB will keep the system under well-

controlled level of memory usage.

Another key issue in this design is splitting VDB into several chunks. Smaller the chunk size, the lower the memory consumption. But smaller sizes require more loading times. If all VDB chunks are of uniform in size, then memory consumption will be better (like dividing logical memory of a program into equal-sized pages in paging technique. The advantage of equal-sized VDB is, when any VDB is removed from the main memory, another new VDB can be completely loaded into the main memory space. Therefore, we can maintain constant memory consumption by the VDB of a scanner.

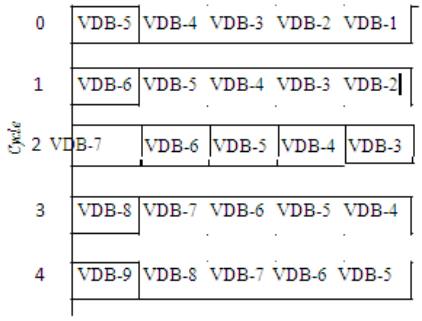


**Fig. 6:** Pipelined loading of VDBs in main memory.

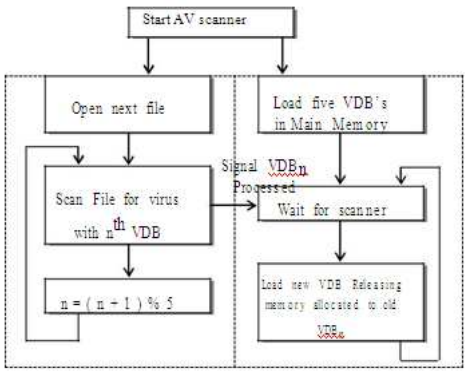


**Fig. 7:** Pipelined scan process.

Fig 7 clearly depicts the working procedure of pipelined scan process. When a user launches the AV scanner, it starts two threads: (1) thread scan engine and (2) thread VDB picker/loader. The thread scan engine responsible for fetching the files from the file system opens it and scans it to detect the virus present with the help of viral patterns in VDB pipeline. Once it completes the scanning with one VDB, immediately it picks the next VDB from the pipeline and it proceeds with the scanning until scanning completed with all VDB chunks. Each time it completes scan with one VDB, it sends a signal complete to the thread VDB picker, so that the latter can remove the recently used VDB from the main memory pipeline and load new one on it.

The thread VDB picker/loader is responsible for loading VDBs from the secondary to the main memory in a pipelined manner. When it is invoked by the AV scanner, it first loads five VDB chunks in the pipeline (here, five VDB chunks can be selected according to user's choice). Then it will wait for the scan engine to complete the scan with any one. VDB(normally scan engine scans with $VDB_1$, $VDB_2$, … $VDB_n$). Once it obtained signal from the scan engine thread, it removes the recently used VDB from the main memory, and all memory space allocated to it also should be released properly. Then, it should load the next VDB chunk in the main memory. The key factor in determining the performance is proper allocation and the release of memory spaces for the VDB chunks throughout the entire scanner's lifetime. Once the scan engine completely scanned the file against all the viral patterns in the VDB chunks, it should immediately fetch the next file from the File system.

Another generally used technique for reducing scan time is excluding some of the files from scanning. For example, files with password protection, image file, and certain system files need not be scanned. This will increase the execution speed considerably.

If the file does not contain any of the known patterns, finally we should perform heuristic analysis to detect the unknown viruses. Most of the antivirus software presently in the market uses this analysis to detect the unidentified viruses. It predicts the potential virus attacks that can happen in future.

The heuristic analysis isolates the file and monitors the file's behavior during execution by allowing it to execute in virtual machine. If the behavior of any file leads to any damage to the system, then those files should be marked as potential virus threat and they should be either cured or deleted or sent to quarantine.

***Conclusion And Future Work:***

In this study, we described a novel approach for designing a scan engine to minimize the memory consumption significantly. Reducing memory consumption means restricting constant/minimal memory allocated to VDB patterns. In order to achieve this, we introduced pipelined VDB, which is the key contribution of this work to minimize memory space consumption. Further improvement in this direction can be achieved by means of parallel scanning of multiple files, which can be in the scope of future studies.

## REFERENCES

Qingguo Wang1,Peilin Jia Zhongming Zhao1, 2013. "VirusFinder: Software for Efficient and Accurate Detection of Viruses and Their Integration Sites in Host Genomes through Next Generation Sequencing Data", 8-5.

Yunxin Chen, Hui Yao, Thompson, E.J., N.M. Tannir, J.N. Weinstein, 2013. VirusSeq: software to identify viruses and their integration sites using nextgeneration sequencing of human cancer tissue. Bioinformatics, 29: 266– 267.

Chieh-Jen Cheng, 2012. "A Scalable high-performance virus detection processor against a Large Pattern Set for Embedded Network Security", 20-5.

Piyachon P. and Y. Luo, 2006. "Efficient memory utilization on network processors for deep packet inspection",presented at the ACM/IEEE Symp Arch. for Netw. Commun. Syst., San Jose, CA.

http://www.tomshardware.com/reviews/anti-virus-virus-sc anner-performance.2777.html.

http://viralpatel.net/taj/tutorial/paging.php.

http://en.wikipedia.org/wiki/Heuristic_analysis.