Reducer : A tool to reduce Redundant Disk I/O

Suhas Aggarwal

Abstract: *Most of the people use virus scanners and security applications such as anti spyware, rootkit hunters to secure their PC. Majority of population also use backup programs to backup their data and prevent data loss. One concern regarding these applications is that they all need to scan the system as part of their functioning. Scanning process demands lots of disk reads, which implies these applications are disk I/O intensive and tend to put a lot of load on the disk. Moreover, when these applications are installed on a single system for different purposes they are meant for, system is read multiple times, one time corresponding to each application, thus inducing redundant Disk I/O.*

In this paper, we create a pipeline to reduce redundant Disk I/O activity and make this process more efficient. We tried to achieve co-ordination between different scanning applications and tried to schedule them using round robin scheme. Trick is to make use of buffer cache to reduce Disk I/O. Accessing buffered file blocks is much cheaper 100-1000 times cheaper than accessing the same block from the disk. When a scanning application does some disk reads, file blocks read will be cached for some time, this fact can be exploited and used to reduce disk I/O, if another scanning application which also need to read these file blocks as part of its functioning, access blocks from buffer cache (cached due to an earlier scanning activity), instead of reading these same blocks again from the disk and thus reducing disk I/O. We achieve this, by scheduling the scans according to a scheduling algorithm namely round robin scheme and have been able to observe significant disk I/O reduction, about 50% in some cases .

Keywords: *Redundant Disk I/O, Filesystem scanning Applications, Round Robin Scheduling, Energy Efficiency, Disk I/O Reduction, Scanning efficiency, Anti-virus software*

INTRODUCTION

Anti-virus software, anti spywares, backup programs have become an integral part of one's life. These are the essential utilities which one must have on his system, to keep his system safe and secure and avoid system failures and downtime. One concern regarding these applications, is that they need to scan the system as part of their functioning. Consequently, these tasks are disk I/O intensive. When installed on a system they induce redundant Disk I/O activity as shown in figure 1.

We aim to reduce redundant Disk I/O activity as shown in figure 2.

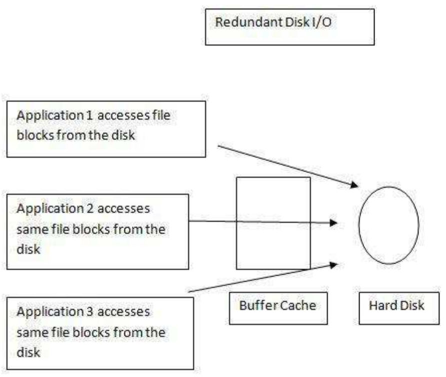

Redundant Disk I/O
Application 1 accesses file blocks from the disk
Application 2 accesses same file blocks from the disk
Application 3 accesses same file blocks from the disk
Buffer Cache
Hard Disk

Figure 1 : Redundant disk I/O

We divided our work in different phases. We conducted some experiments to study the behavior of malware checkers involves trapping file system calls, monitoring I/O activity, doing scan analysis like time to do a full system scan, types of files being read by these scanners.

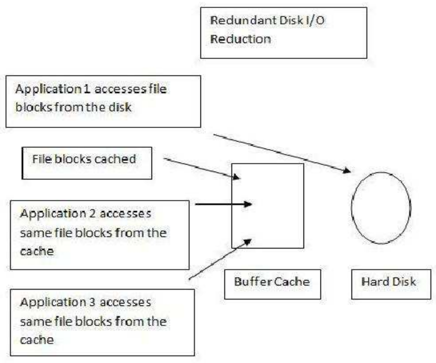

Redundant Disk I/O Reduction
Application 1 accesses file blocks from the disk
File blocks cached
Application 2 accesses same file blocks from the cache
Application 3 accesses same file blocks from the cache
Buffer Cache
Hard Disk

Figure 2: Redundant disk I/O Reduction

After that we designed and implemented a scheduling service which will work to co-ordinate different scanning applications. We were able to implement a friendly API to which scanners can be interfaced with no scanner code editing requirements. Finally, we conducted experiment t o demonstrate reduced disk I/O.

1. Main Principle

Main concept behind our implementation is to make use of file system caching. Disk accesses are very expensive. When a file is read its blocks are buffered in the cache for a while. Accessing a buffered block is much cheaper about (100-1000 times cheaper) than accessing the same block from the disk, we exploited this difference. When a set of files is read, we give every tool an opportunity to scan it, if it wishes to. This is the cheapest time to do the scan because disk blocks are likely to be in the c ache. We implemented a prototype which is a schedular service which runs to co-ordinate the scans and reduce redundant Disk I/O activity.

1.1 Scheduling the scans with a round robin scheme

1.2 Scan scheduling criteria.
Queue - tool1--->tool2--->tool3

Each tool in the queue does the scan for a fixed time quanta and is able to complete its scan in certain no. of rounds. At the end of a time quanta, tool doing the scan saves its state, next tool in the queue is given the opportunity to scan. In the next round, tool starts from the state it left and cycle continues till it has scanned the entire file.

File blocks read by tool1 in first round are cached, tool 2 tends to read the same file blocks when it starts, but it fetches these block from the buffer cache instead from the disk, thus saving expensive disk I/O operations. Same procedure is followed by tool3.

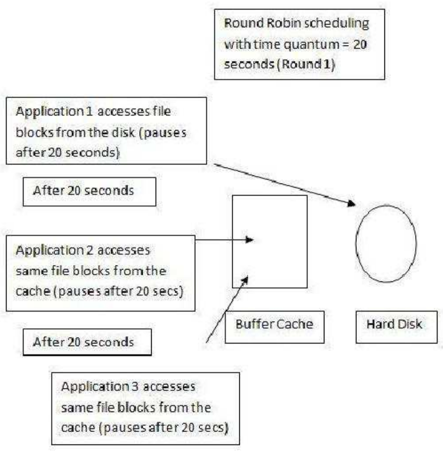

Round Robin scheduling with time quantum = 20 seconds (Round 1)
Application 1 accesses file blocks from the disk (pauses after 20 seconds)
After 20 seconds
Application 2 accesses same file blocks from the cache (pauses after 20 secs)
After 20 seconds
Application 3 accesses same file blocks from the cache (pauses after 20 secs)
Buffer Cache
Hard Disk

Figure 3: Round robin scheduling (round 1)

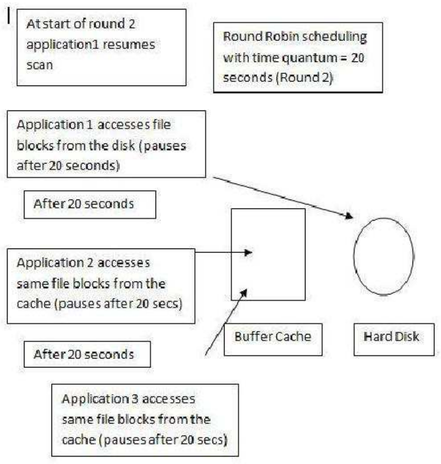

At start of round 2 application1 resumes scan
Round Robin scheduling with time quantum = 20 seconds (Round 2)
Application 1 accesses file blocks from the disk (pauses after 20 seconds)
After 20 seconds
Application 2 accesses same file blocks from the cache (pauses after 20 secs)
After 20 seconds
Application 3 accesses same file blocks from the cache (pauses after 20 secs)
Buffer Cache
Hard Disk

Figure 4: Round robin scheduling (round 2)

We implemented schedular service and studied impact of quantum size as first step

2. Reducer

Reducer is scheduling service to make scanning applications run according to a scheduling algorithm.

2.1 Reducer design

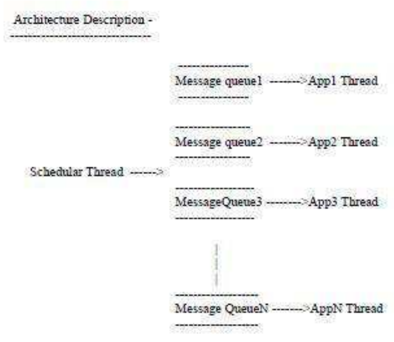

Architecture Description -
Message queue1 ------->App1 Thread
Message queue2 ------->App2 Thread
Schedular Thread ----->
MessageQueue3 -------->App3 Thread
Message QueueN ------->AppN Thread

Figure 5: Architecture design

There are N application threads, each corresponding to scanning application. Thread poll their respective message queues continuously, looking for messages. Schedular thread writes messages in message queue. Schedular thread writes the appropriate function code in the message queue, application thread read it from the message queue and perform respective function. There are 3 function codes - 100->start the scan,101->pause the scan,110->resume the paused scan

Please note that a synchronization mechanism will be needed when Schedular thread writes to the message queue and Application thread reads from the message queue.

A sample run

There are 3 scanning applications. Time quanta = 20 seconds. At the start of the scan, 1st round of round robin - Schedular thread writes 100 in message queue 1, application 1 thread reads this message and scan by first application starts. After 20 seconds (end of first time quantum), schedular thread writes 101 in message queue 1 and 100 in message queue 2, application threads read their respective message queues, application 1 pauses and application 2 starts the scan. At the end of 20 seconds, (end of second time quantum), schedular thread writes 101 in message queue 2 and 100 in message queue 3, application 2 pauses its scan, application 3 starts its scan. After 20 seconds, first round completes. Now 1st round of round robin has completed. At the start of 2nd round, schedular thread writes 110 in message queue 1 and 101 in message queue 3, meaning resume the paused application 1 and and pause application 3. After 20 seconds(when first time quantum of second round expires) schedular thread writes 101 in message queue 1, and 110 in message queue 2. Application 1 pauses its scan and application 2 resumes its scan. Similarly, at the end of time quantum, application 2 pauses its scan and application 3 resumes its scan.

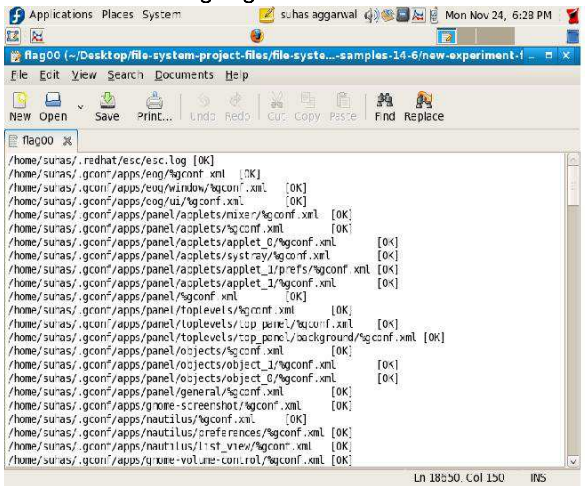
3. Schedular experiment

3.1 Studying Disk I/O Activity when Disk I/O intensive applications work individually (as when during a normal periodic scan)

For simplicity, we are studying the disk I/O caused by scanning applications of the same type – virus scanners at first. We consider three virus scanners and study Disk I/O activity when each of them work individually (as they do during normal periodic scan). We will also study the disk I/O redundancy three scanners induce, and reduction in disk I/O redundancy when three scanners are scheduled using a scheduling algorithm, particularly round robin scheme, which we have chosen.

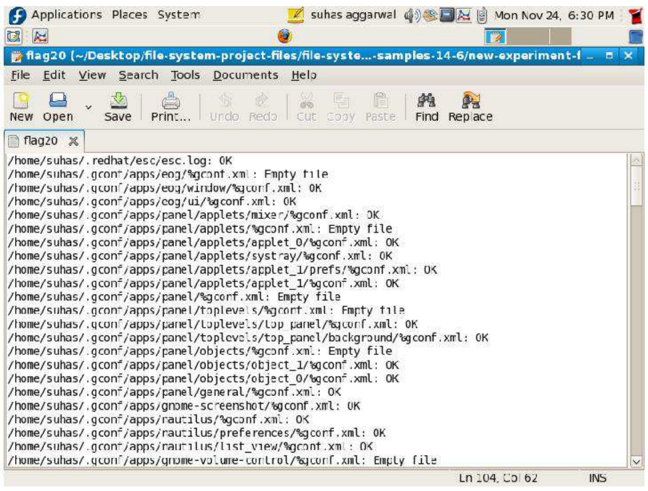
Experiment was conducted on Fedora core 12 system with 1 GB RAM. Three virus scanners Avast, ClamAV, Vexira antivirus were used. Iostat tool was used to monitor Disk I/O activity.

File scanning logs of different virus scanners –


Applications Places System
suhas aggarwal Mon Nov 24, 6:28 PM
flag00 (~/Desktop/file-system-project-files/file-syste...-samples-14-6/new-experiment-f
File Edit View Search Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
flag00
Ln 18550, Col 150 INS

```
/home/suhas/.redhat/esc/esc.log [OK]
/home/suhas/.gconf/apps/eog/%gconf.xml  [OK]
/home/suhas/.gconf/apps/eog/window/%gconf.xml   [OK]
/home/suhas/.gconf/apps/eog/ui/%gconf.xml       [OK]
/home/suhas/.gconf/apps/panel/applets/mixer/%gconf.xml  [OK]
/home/suhas/.gconf/apps/panel/applets/%gconf.xml        [OK]
/home/suhas/.gconf/apps/panel/applets/applet_0/%gconf.xml       [OK]
/home/suhas/.gconf/apps/panel/applets/systray/%gconf.xml        [OK]
/home/suhas/.gconf/apps/panel/applets/applet_1/prefs/%gconf.xml [OK]
/home/suhas/.gconf/apps/panel/applets/applet_1/%gconf.xml       [OK]
/home/suhas/.gconf/apps/panel/%gconf.xml        [OK]
/home/suhas/.gconf/apps/panel/toplevels/%gconf.xml      [OK]
/home/suhas/.gconf/apps/panel/toplevels/top_panel/%gconf.xml    [OK]
/home/suhas/.gconf/apps/panel/toplevels/top_panel/background/%gconf.xml [OK]
/home/suhas/.gconf/apps/panel/objects/%gconf.xml        [OK]
/home/suhas/.gconf/apps/panel/objects/object_1/%gconf.xml       [OK]
/home/suhas/.gconf/apps/panel/objects/object_0/%gconf.xml       [OK]
/home/suhas/.gconf/apps/panel/general/%gconf.xml        [OK]
/home/suhas/.gconf/apps/gnome-screenshot/%gconf.xml     [OK]
/home/suhas/.gconf/apps/nautilus/%gconf.xml     [OK]
/home/suhas/.gconf/apps/nautilus/preferences/%gconf.xml [OK]
/home/suhas/.gconf/apps/nautilus/list_view/%gconf.xml   [OK]
/home/suhas/.gconf/apps/gnome-volume-control/%gconf.xml [OK]
```

Avast virus scanner log


Applications Places System
suhas aggarwal Mon Nov 24, 6:30 PM
flag20 (~/Desktop/file-system-project-files/file-syste...-samples-14-6/new-experiment-f
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
flag20
Ln 104, Col 62 INS

```
/home/suhas/.redhat/esc/esc.log: OK
/home/suhas/.gconf/apps/eog/%gconf.xml: Empty file
/home/suhas/.gconf/apps/eog/window/%gconf.xml: OK
/home/suhas/.gconf/apps/eog/ui/%gconf.xml: OK
/home/suhas/.gconf/apps/panel/applets/mixer/%gconf.xml: OK
/home/suhas/.gconf/apps/panel/applets/%gconf.xml: Empty file
/home/suhas/.gconf/apps/panel/applets/applet_0/%gconf.xml: OK
/home/suhas/.gconf/apps/panel/applets/systray/%gconf.xml: OK
/home/suhas/.gconf/apps/panel/applets/applet_1/prefs/%gconf.xml: OK
/home/suhas/.gconf/apps/panel/applets/applet_1/%gconf.xml: OK
/home/suhas/.gconf/apps/panel/%gconf.xml: Empty file
/home/suhas/.gconf/apps/panel/toplevels/%gconf.xml: Empty file
/home/suhas/.gconf/apps/panel/toplevels/top_panel/%gconf.xml: OK
/home/suhas/.gconf/apps/panel/toplevels/top_panel/background/%gconf.xml: OK
/home/suhas/.gconf/apps/panel/objects/%gconf.xml: Empty file
/home/suhas/.gconf/apps/panel/objects/object_1/%gconf.xml: OK
/home/suhas/.gconf/apps/panel/objects/object_0/%gconf.xml: OK
/home/suhas/.gconf/apps/panel/general/%gconf.xml: OK
/home/suhas/.gconf/apps/gnome-screenshot/%gconf.xml: OK
/home/suhas/.gconf/apps/nautilus/%gconf.xml: OK
/home/suhas/.gconf/apps/nautilus/preferences/%gconf.xml: OK
/home/suhas/.gconf/apps/nautilus/list_view/%gconf.xml: OK
/home/suhas/.gconf/apps/gnome-volume-control/%gconf.xml: Empty file
```

ClamAV virus scanner log

It can be observed from file scanning logs of virus scanners above that different virus scanners employ same file scanning algorithm. This is also the requirement for our prototype, file scanning applications should scan the files in the same order. Individual Disk I/O different virus scanners were inducing. File system size – 1.8GB, Avast – 1.4GB, Vexira- 1.35GB, Clamav – 1.5GB.

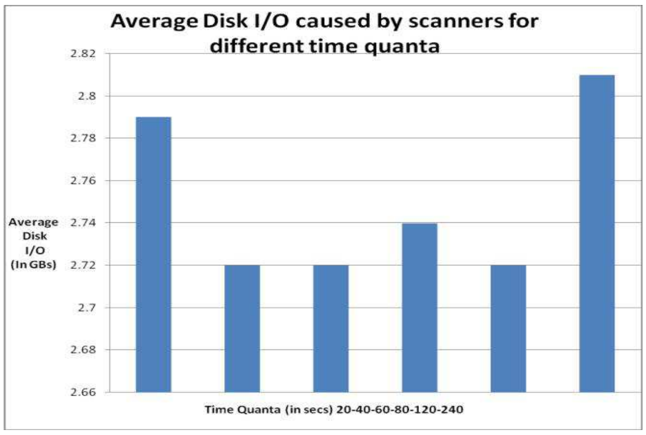
3.2 Studying Disk I/O activity when 3 scanning applications are scheduled with a scheduling algorithm namely round robin scheme

We study disk I/O activity when three scanning application i.e virus scanners are scheduled according to round robin scheme. Experiment was conducted on Fedora core 12 system with 1 GB RAM. Three virus scanners Avast, ClamAV, Vexira antivirus were used. Iostat tool was used to monitor Disk I/O activity. Experiment was conducted for time quanta t=20, 40, 60, 80,120 and 240 seconds.

Experiment Results –

Time Quanta (in secs)	20	40	60	80	120	240
DiskI/O (Sample1) (in GBs)	2.77	2.71	2.73	2.73	2.75	2.85
DiskI/O (Sample2) (in GBs)	2.79	2.73	2.71	2.73	2.66	2.71
DiskI/O (Sample3) (in GBs)	2.81	2.72	2.73	2.76	2.74	2.88
Average	2.79	2.72	2.72	2.74	2.72	2.81

Table 1

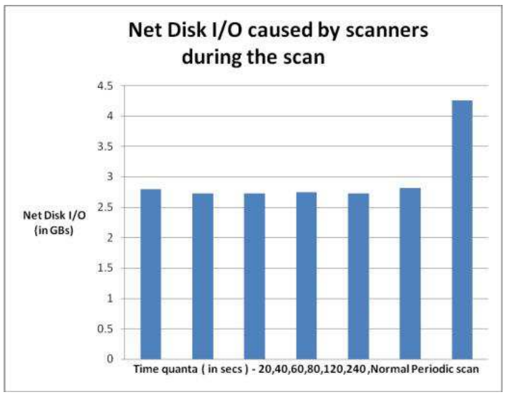


One can observe net average Disk I/O caused was lowest for t=40 & 60 seconds and highest for t= 240 seconds.

3.3. Net Disk I/O caused by scanners in different cases

When scheduled individually (equal to net Disk I/O for a normal periodic scan) Net Disk I/O= Net Disk I/O of (Avast Individual scan + Vexira Individual scan + Clamscan Individual scan) = (1. 4 + 1.35 + 1.5)GB = 4.25GB.
Filesystem size = 1.8GB
Redundant Disk I/O = (4.25- 1.8)GB=2.45GB
Net Disk I/O caused by scanners when they are scheduled according to a round robin scheme using Scheduler.
Scans were conducted for three different time quanta values. Some statistics -
1)Net Disk I/O for t=20sec=2.79GB. Redundant Disk I/O=(2.79-1.8)GB=0.99GB.
2)Net Disk I/O for t=40sec=2.72GB. Redundant Disk I/O=(2.72-1.8)GB=0.92GB.
3)Net Disk I/O for t=60sec=2.72GB. Redundant Disk I/O=(2.72- 1. 8)GB=0. 92GB.
4)Net Disk I/O for t=80sec=2.74GB. Redundant Disk I/O=(2.74–1.8)GB=0.94GB.
5)Net Disk I/O for t=120sec=2.72GB. Redundant Disk I/O=(2.72 -1.8)GB=0.92GB.
6)Net Disk I/O for t=240sec=2.81GB. Redundant Disk I/O=(2.81 -1.8)GB=1.01GB.

One can observe significant reduction in redundant data, drops from 2.45 GB to 0.92GB (for time quanta t= 40,60,120 seconds) and 0.94 G, (time quanta t=80 seconds).



4. Application : Fast virus checking

Anti-virus software, anti spywares, backup programs have become an integral part of one's life. These are the essential utilities which one must have on his system, to keep his system safe and secure and avoid system failures and downtime. One concern regarding these applications, are that they need to scan the system as part of their functioning. Consequently, these tasks are disk I/O intensive.

People often experience slow downs when these applications are running, firstly because these tasks are Disk I/O intensive and secondly as they follow figure 1 below, some aware people follow figure 2. But, as these tasks need to be performed regularly, they often become a problem for the people and they tend to avoid it. With the help of Reducer ,we can make the process of scanning more efficient, by reducing redundant disk I/O activity, reducing downtime (time when scans are being conducted by these applications.).

Principle:

Reducer works on the principle, first application reads file blocks from the disk and later application which also need access to these file blocks, access them from buffer cache, rather than reading from the disk. As accessing file blocks from buffer cache, is much faster than accessing file blocks from the disk, applications are able to read large no. of file blocks in less time and their scanning efficiency improves as compared to when they were reading file blocks from the disk.

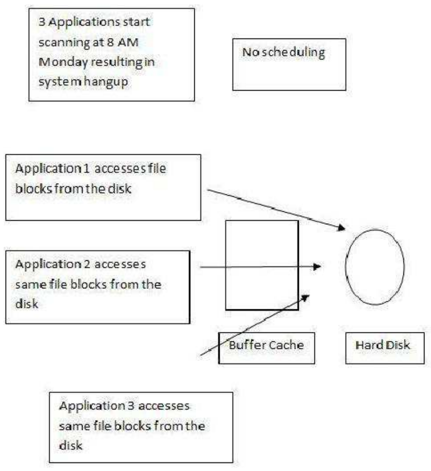

3 Applications start scanning at 8 AM Monday resulting in system hangup
No scheduling
Application 1 accesses file blocks from the disk
Application 2 accesses same file blocks from the disk
Buffer Cache
Hard Disk
Application 3 accesses same file blocks from the disk

Figure 1

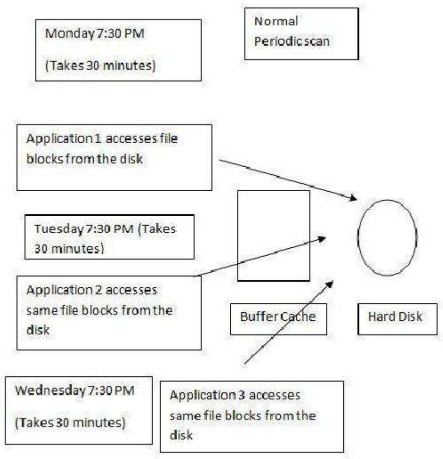

Monday 7:30 PM (Takes 30 minutes)
Normal Periodic scan
Application 1 accesses file blocks from the disk
Tuesday 7:30 PM (Takes 30 minutes)
Application 2 accesses same file blocks from the disk
Buffer Cache
Hard Disk
Wednesday 7:30 PM (Takes 30 minutes)
Application 3 accesses same file blocks from the disk

Figure 2

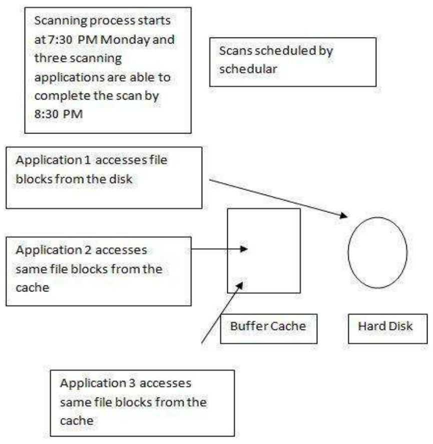

Scanning process starts at 7:30 PM Monday and three scanning applications are able to complete the scan by 8:30 PM
Scans scheduled by schedular
Application 1 accesses file blocks from the disk
Application 2 accesses same file blocks from the cache
Buffer Cache
Hard Disk
Application 3 accesses same file blocks from the cache

Figure 3

Figure 3 illustrates, how we can improve scanning efficiency using Reducer.

As one can observe, one will be able to complete the three scans in a single day in approximately half the time.

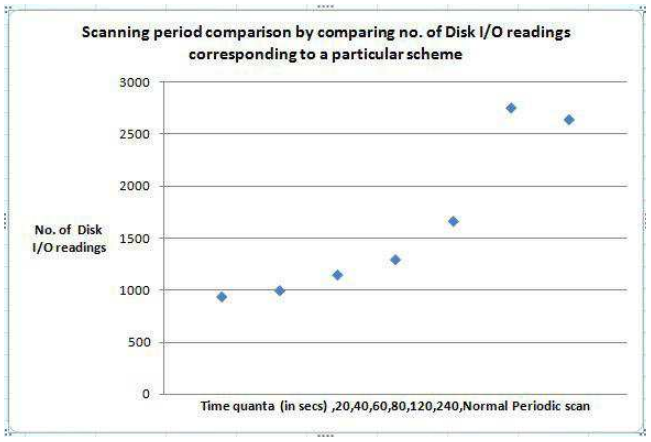
5. Time spent in scanning

We can use the no. of Disk I/O readings (those corresponding to disk reads by scanners)in a scan sheet as a measure of scanning period (defined between non idle reading slot s (idle slots correspond to period when kr/s was zero ,scan was about to begin or has ended.))When iostat will trap non -null disk reads (those which are done by scanner while scanning),it implies scan is still in progress so larger are no. of disk reads ,larger is the scanning period. So, we conclude that No. of Disk I/O readings in a scan sheet corresponding to a particular scheme are directly proportional to time spent in scanning, when that particular scheme was employed for scheduling scans. Suppose, if round robin scheme with time quanta t=20 has less no. of Disk I/O readings (corresponding to disk reads by scanners)in its scan sheet as compared to time quanta t=60, we can safely, say that former scheme takes less time to scan as compared to latter.

No. of Disk I/O readings obtained for different schemes –

Normal Periodic scan
1)Avast- 744
2)Vascan - 495
3)ClamAV - 1403
Time quanta t=20 – 939
Time quanta t=40 - 999
Time quanta t=60 – 1149
Time quanta t=80 – 1296
Time quanta t=120 - 1665
Time quanta t=240 – 2754

As, can be observed, scanning time reduces to half when time quanta is set to 20, 40, 60 seconds. So, we can achieve our scenario (fig 3) if time quanta is set to 20, 40, 60 seconds. One can schedule the scans using reducer, on Monday 7:30 PM, and can experience performance gain.



6. Discussion

It can be argued that paper shows scheduling of three virus scanners according to a round robin scheme which is not the practical case, but this exercise is just for the proof of concept, in practice three scanning applications of different nature will be scheduled. One more important point can be that it is argued that I/O intensive daemons run infrequently on a system but above system make them run concurrently. It will be good to study how this problem affects user experience. Also, there can be a point regarding, how sensitive is performance improvements corresponding to system noise and other background load which is inevitable in daemon-rich environments.

7. Conclusion

Finally we conclude that we that we can reduce Disk I /O redundancy using reducer. It can be used for fast virus checking, as we have shown earlier that we were able to complete the full scan by three virus scanners in half the time for time quantum setting of t = 20, 40 and 60 seconds as compared to a normal periodic time. Thus it helps in reducing downtime.

In future, we aim to study integration of Reducer at kernel level. We also aim to study impact of disk read, categorize processes which can be executed while scans are going on, without any slowdowns. Finally, we also plan to do study of a practical case when virus scanner, a security application like Microsoft anti spyware and a back up program are scheduled using reducer and do performance analysis. One more interesting study can be what effect reducer has on the power consumption of the device. Does it lead to an Energy Efficient system?

8. Acknowledgements

I would like to thank Dr. Atul Prakash and Dr. Hemangee Kapoor for providing immense guidance and support.

REFERENCES

[1]A New Proportional-Share Disk Scheduling Algorithm: Trading-off I/O Throughput and QoS guarantee. *Lecture Notes in Computer Science (PIOMT2003)*, pp. 257-266, June 2003. By Young Jin Nam and Chanik Park.

[2]Improving Disk Performance Via Latency Reduction. IEEE *Transactions on Computers* Volume 40 , Issue 1 (January 1991) Pages: 22 - 30

[3]High-performance disk I/O in a bus-based system. Starkville, Mississippi, J.C. Harden, D.H. Linder, S. Kadambi, *SSST'95 Proceedings of the 27th Southeastern Symposium on System Theory*

[4]Real -time disk scheduling algorithm allowing c oncurrent I/O requests . Staelin, Carl; Amir, Gidi; Ben-Ovadia, David; Dagan, Ram; Melamed, Michael; Staas, Dave HP Laboratories *HPL-2009-344*

[5]Improving Disk I/O Performance by Using Raw Disk for Web Proxy Servers Jong -IkShim, Jae-DongLee *HSI'03 Proceedings of the 2nd international conference on Human society@internet*

[6]Boosting Performance for I/O -Intensive Workload by Preemptive Job Migrations in a Cluster System
Proceedings of the 15th Symposium on Computer Architecture and High Performance Computing.

[7] K.S. Grimsmond, J.K. Archibald, B.E. Nelson, Multiple prefetch adaptive disk caching*, IEEE Trans actions on Knowledge and data Engineering 5 (1) (1993) 88–103.*

[8]Operating System Concepts: Seventh Edition, Avi Silberschatz Peter Baer Galvin Greg Gagne

About Author:

Suhas Aggarwal holds a B.Tech in computer science from IIT Guwahati. He currently works as a software engineer at Times Internet Ltd. He can be reached at – suhasagg@gmail.com