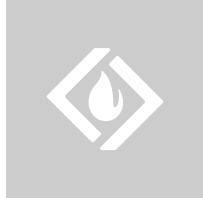


[/ Home](#) / [Browse](#) / [virgo](#) Code

virgo

32 bit VIRGO Linux Kernel

Brought to you by: ka_shrinivaasan

[r844]:  / [trunk](#) / [virgo-docs](#) / VirgoDesign.txt

 Restore 

History

[Download this file](#)

1203 lines (960 with data), 100.6 kB

```
1  /*****
2  #------
3  #NEURONRAIN VIRGO - Cloud, Machine Learning and Queue augmented Linux Kernel Fork-off
4  #This program is free software: you can redistribute it and/or modify
5  #it under the terms of the GNU General Public License as published by
6  #the Free Software Foundation, either version 3 of the License, or
7  #(at your option) any later version.
8  #This program is distributed in the hope that it will be useful,
9  #but WITHOUT ANY WARRANTY; without even the implied warranty of
10 #MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
11 #GNU General Public License for more details.
12 #You should have received a copy of the GNU General Public License
13 #along with this program. If not, see <http://www.gnu.org/licenses/>.
14 #------
15 #Copyleft (Copyright+):
16 #Srinivasan Kannan
17 #(also known as: Ka.Shrinivaasan, Shrinivas Kannan)
```

```

18 #Ph: 9791499106, 9003082186
19 #Krishna iResearch Open Source Products Profiles:
20 #http://sourceforge.net/users/ka_shrinivaasan,
21 #https://github.com/shrinivaasanka,
22 #https://www.openhub.net/accounts/ka_shrinivaasan
23 #Personal website(research): https://sites.google.com/site/kuja27/
24 #emails: ka.shrinivaasan@gmail.com, shrinivas.kannan@gmail.com,
25 #kashrinivaasan@live.com
26 #-----
27 *****/
28
29 VIRGO is an operating system kernel forked off from Linux kernel mainline to add cloud functionalities (system c
30
31 Remote Device Invocation , which is an old terminology for Internet-Of-Things has already been experimented in Sun
32
33 Memory pooling:
34 -----
35 Memory pooling is proposed to be implemented by a new virgo_malloc() system call that transparently allocates a l
36
37 CPU pooling or cloud ability in a system call:
38 -----
39 Clone() system call is linux specific and internally it invokes sys_clone(). All fork(),vfork() and clone() syste
40
41 virgo_clone() is a wrapper over clone() that looks up a map of machines-to-loadfactor and get the host with leas
42
43 Kernel has support for kernel space sockets with kernel_accept(), kernel_bind(), kernel_connect(), kernel_sendms
44
45 Experimental Prototype
46 -----
47 virgo_clone() system call and a kernel module virgocloudexec which implements Sun RPC interface have been implem
48
49 VIRGO - loadbalancer to get the host:ip of the least loaded node
50 -----
51 Loadbalancer option 1 - Centralized loadbalancer registry that tracks load:
52 -----
53
54 Virgo_clone() system call needs to lookup a registry or map of host-to-load and get the least loaded host:ip from
55
56 Many application level userspace load monitoring tools are available but as virgo_clone() is in kernel space, it
57

```

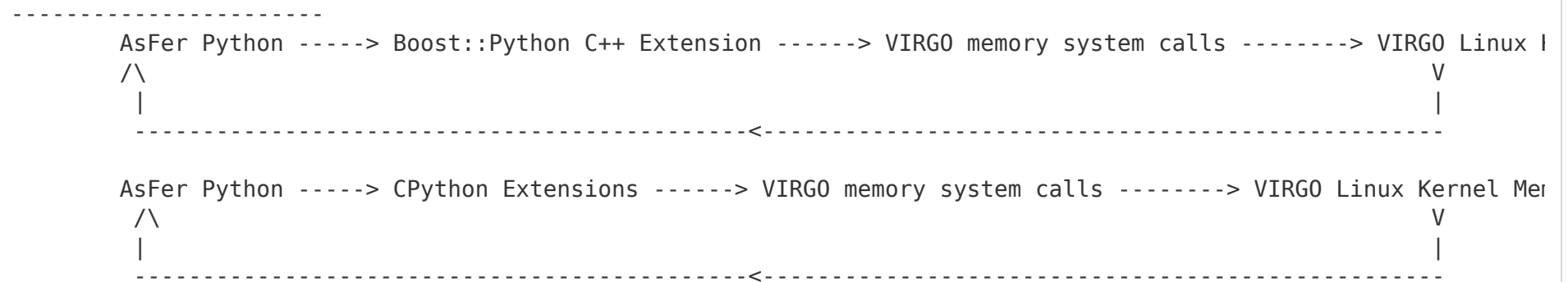
```

58 (Design notes for LB option 1 handwritten by myself are at :http://sourceforge.net/p/virgo-linux/code-0/HEAD/tree/
59
60 Loadbalancer option 2 - Linux Psuedorandom number generator based load balancer(experimental) instead of central:
61 -----
62
63 Each virgo_clone() client has a PRG which is queried (/dev/random or /dev/urandom) to get the id of the host to :
64 Expected number of requests per node is derived as:
65
66 expected number of requests per node = summation(each_value_for_the_random_variable_for_number_of_requests * prob
67
68 =expected number of requests per node = (math.pow(N, k+2) - k*math.pow(N,2) + k*math.pow(N,1) - 1) / (math.pow(N
69
70 This loadbalancer is dependent on efficacy of the PRG and since each request is uniformly, identically, independe
71 would distribute requests evenly. This obviates the need for loadtracking and coherency of the load-to-host tabl
72
73 (Design notes for LB option 2 handwritten by myself at :http://sourceforge.net/p/virgo-linux/code-0/HEAD/tree/tri
74
75
76 (python script in virgo-python-src/)
77
78 *****
79 Implemented VIRGO Linux components (as on 7 March 2016)
80 *****
81 1. cpupooling virtualization - VIRGO_clone() system call and VIRGO cpupooling driver by which a remote procedure
82 2. memorypooling virtualization - VIRGO_malloc(), VIRGO_get(), VIRGO_set(), VIRGO_free() system calls and VIRGO
83 3. filesystem virtualization - VIRGO_open(), VIRGO_read(), VIRGO_write(), VIRGO_close() system calls and VIRGO c
84 4. config - VIRGO config driver for configuration symbols export.
85 5. queueing - VIRGO Queueing driver kernel service for queuing incoming requests, handle them with workqueue and :
86 6. cloudsync - kernel module for synchronization primitives (Bakery algorithm etc.,) with exported symbols that
87 7. utils - utility driver that exports miscellaneous kernel functions that can be used across VIRGO Linux kernel
88 8. EventNet - eventnet kernel driver to vfs_read()/vfs_write() text files for EventNet vertex and edge messages
89
90 9. Kernel_Analytics - kernel module that reads machine-learnt config key-value pairs set in /etc/virgo_kernel_an
91 10. Testcases and kern.log testlogs for the above
92 11. SATURN program analysis wrapper driver.
93
94 Thus VIRGO Linux at present implements a minimum cloud OS (with cloud-wide cpu, memory and file system managemen
95
96 *****
97 VIRGO ToDo and NiceToHave Features (list is quite dynamic and might be rewritten depending on feasibility - long
98 *****

```

(FEATURE - DONE-minimum separate config file support in client and kernel service)1. More Sophisticated VIRGO c
 (FEATURE - Special case implementation DONE) 2. Object Marshalling and Unmarshalling (Serialization) Features - I
 (FEATURE - DONE) 3. Virgo_malloc(), virgo_set(), virgo_get() and virgo_free() syscalls that virtualize the physi
 Initial Design Handwritten notes committed at: <http://sourceforge.net/p/virgo-linux/code-0/210/tree/trunk/virgo->
 (FEATURE - DONE) 4. Integrated testing of AsFer-VIRGO Linux Kernel request roundtrip - invocation of VIRGO linux

4.1 Schematic Diagram:



(FEATURE - DONE)5. Multithreading of VIRGO clouDEXEC kernel module (if not already done by kernel module subsystem)
 (FEATURE - DONE) 6. Sophisticated queuing and persistence of CPU and Memory pooling requests in Kernel Side (by I
 (FEATURE - DONE-Minimum Functionality) 7. Integration of Asfer(AstroInfer) algorithm codes into VIRGO which would

 Example scenario 1 without implementation:

- Philips Hue IoT mobile app controlled bulb - <http://www2.meethue.com/en-xx/>
- kernel_analytics module learns key-value pairs from the AsFer code and exports it VIRGO kernel wide
- A driver function with in bulb embedded device driver can be invoked through VIRGO cpupooling (invoked from re
 based on if-else clause of the kernel_analytics variable i.e remote_client invokes virgo_clone() with function a

 Example scenario 2 without implementation:

- A swivel security camera driver is remotely invoked via virgo_clone() in the VIRGO cloud.
- The camera driver uses a machine learnt variable exported by kernel_analytics-and-AsFer to pan the camera by h

```
-----
138 Example scenario 3 without implementation - probably one of the best applications of NeuronRain IoT OS:
139 -----
140
141 - Automatic Driverless Automobiles - a VIRGO driver for a vehicle which learns kernel analytics variables (drivin
142   - AsFer analytics receives obstacle distance data 360+360 degrees around (horizontal and vertical) the ve
143   - VIRGO Linux kernel on vehicle has two special drivers for Gear-Clutch-Break-Accelerator-Fuel(GCBAF) and
144   - AsFer analytics with high frequency computes threshold variables for applying break, clutch, gear, vel
145   - These analytics variables are continuously read by GCBAF and Steering drivers which autopilot the vehi
146   - Above applies to Fly-by-wire aeronautics too with appropriate changes in analytics variables computed.
147   - The crucial parameter is the response time in variable computation and table updates which requires a l
148
149 -----
150 References for Machine Learning + Linux Kernel
151 -----
152 7.1 KernTune - http://repository.uwc.ac.za/xmlui/bitstream/handle/10566/53/Yi\_KernTune\(2007\).pdf?sequence=3
153 7.2 Self-learning, Predictive Systems - https://icri-ci.technion.ac.il/projects/past-projects/machine-learning-f
154 7.3 Linux Process Scheduling and Machine Learning - http://www.cs.ucr.edu/~kishore/papers/tencon.pdf
155 7.4 Network Latency and Machine Learning - https://users.soe.ucsc.edu/~slukin/rtt\_paper.pdf
156 7.5 Machine Learning based Meta-Scheduler for Multicore processors - https://books.google.co.in/books?id=1GWcHmC
157
158 8. A Symmetric Multi Processing subsystem Scheduler that virtualizes all nodes in cloud (probably this would inv
159
160 (FEATURE - ONGOING) 9. Virgo is an effort to virtualize the cloud as a single machine - Here cloud is not limite
161
162 (FEATURE - DONE) 10. Memory Pooling Subsystem Driver - Virgo_malloc(), Virgo_set(), Virgo_get() and Virgo_free()
163
164 (FEATURE - DONE) 11. Virgo Cloud File System with virgo_cloud_open(), virgo_cloud_read() , virgo_cloud_write() a
165
166 (FEATURE - DONE) 12. VIRGO Cloud File System commands through syscall paths - virgo_open(),virgo_close(),virgo_r
167
168 (FEATURE - DONE) 13. VIRGO memory pooling feature is also a distributed key-value store similar to other promine
169
170 14. VIRGO memory pooling can be improved with disk persistence for in-memory key-value store using virgo_malloc(
171
172 15. (FEATURE-DONE) Socket Debugging, Program Analysis and Verification features for user code that can find bugs
173
174 16(FEATURE - DONE-Minimum Functionality). Operating System Logfile analysis using Machine Learning code in Astro:
175
176 17. Implementations of prototypical Software Transactional Memory and LockFree Datastructures for VIRGO memory p
177
```

18. Scalability features for Multicore machines - references:
(<http://halobates.de/lk09-scalability.pdf>, <http://pdos.csail.mit.edu/papers/linux/osdil0.pdf>)

19. Read-Copy-Update algorithm implementation for VIRGO memory pooling that supports multiple simultaneous versions

20. (FEATURE - SATURN integration - minimum functionality DONE) Program Comprehension features as an add-on description

21. (FEATURE - DONE) Bakery Algorithm implementation - cloudsync kernel module

22. (FEATURE - ONGOING) Implementation of Distributed Systems primitives for VIRGO cloud viz., Logical Clocks, Total Order

23. (FEATURE - minimum functionality DONE) Enhancements to kmem if it makes sense, because it is better to rely on it
Kernel Malloc syscall kmalloc() internally works as follows:

- kmem_cache_t object has pointers to 3 lists
- These 3 lists are full objects SLAB list, partial objects SLAB list and free objects SLAB list - all are in cache
- and cache_cache is the global list of all caches created thus far.
- Any kmalloc() allocation searches partial objects SLAB list and allocates a memory block with kmem_cache_alloc()
- Any kfree() returns an object to a free SLAB list
- Full SLABs are removed from partial SLAB list and appended to full SLAB list
- SLABs are virtual memory pages created with kmem_cache_create
- Each SLAB in SLABs list has blocks of similar sized objects (e.g. multiples of two). Closest matching block is used

KERNELSPACE:

VIRGO address translation table already implements a tree registry of vtables each of capacity 3000 that keep track of virtual addresses

USERSPACE: sbrk() and brk() are no longer used internally in malloc() library routines. Instead mmap() has replaced them

24.(FEATURE - ONGOING) Cleanup the code and remove unnecessary comments.

25.(FEATURE - DONE) Documentation - This design document is also a documentation for commit notes and other build system

26. (FEATURE - DONE) Telnet path to virgo_cloud_malloc,virgo_cloud_set and virgo_cloud_get has been tested and works

27. Augment the Linux kernel workqueue implementation (<http://lxr.free-electrons.com/source/kernel/workqueue.c>) with a new workqueue

28.(FEATURE - DONE) VIRGO queue driver with native userspace queue and kernel workqueue-handler framework that is used by the

29.(FEATURE - DONE) KERNELSPACE EXECUTION ACROSS CLOUD NODES which geographically distribute userspace and kernel execution

a logical abstraction for a cloudwide virtualized kernel:

Remote Cloud Node Client

```

218         (cpupooling, eventnet, memorypooling, cloudfs, queueing - telnet and syscalls clients)
219         |
220         |
221     (Userspace) |
222         |-----Kernel Sockets-----> Remote
223                                         (VIRGO cpupooling, memorypooling, cloudf
224
225
226
227
228
229         <-----Kernel Sockets-----
230         |
231         |
232         |
233     (Userspace) |
234
235
236

```

30. (FEATURE - DONE) VIRGO platform as on 5 May 2014 implements a minimum set of features and kernelsocket comman

31. (FEATURE - DONE) VIRGO Queue standalone kernel service has been implemented in addition to paths in schemati

VIRGO Queue client(e.g telnet) -----> VIRGO Queue kernel service ---> Linux Workqueue handler -----> KingCobra

32. (FEATURE - DONE) EventNet kernel module service:

VIRGO eventnet client (telnet) -----> VIRGO EventNet kernel service -----> EventNet graph text files

33. (FEATURE - DONE) Related to point 22 - Reuse EventNet cloudwide logical time infinite graph in AsFer in plac

34. (FEATURE - OPTIONAL) The kernel modules services listening on ports could return a JSON response when connec

35. (FEATURE-Minimum Functionality DONE) Pointer Swizzling and Unswizzling of VIRGO addressspace pointers to/from

CODE COMMIT RELATED NOTES

VIRGO code commits as on 16/05/2013

```
258 -----
259 1. VIRGO clouddriver with a listener kernel thread service has been implemented and it listens on port 10000
260 through /etc/modules load-on-bootup facility
261
262 2. VIRGO clouddriver virgo_clone() system call has been implemented that would kernel_connect() to the VIRGO cloud
263 port 10000
264
265 3. VIRGO clouddriver has been split into virgo.h (VIRGO typedefs), virgocloudexecsvc.h(VIRGO clouddriver serv
266 module_init() of VIRGO clouddriver) and virgo_cloudexec.c (with module ops definitions)
267
268 4. VIRGO does not implement SUN RPC interface anymore and now has its own virgo ops.
269
270 5. Lot of Kbuild related commits with commented lines for future use have been done viz., to integrate VIRGO to I
271
272 VIRGO code commits as on 20/05/2013
273 -----
274 1. test_virgo_clone.c testcase for sys_virgo_clone() system call works and connections are established to VIRGO
275
276 2. Makefile for test_virgo_clone.c and updated buildscript.sh for headers_install for custom-built linux.
277
278 VIRGO code commits as on 6/6/2013
279 -----
280 1. Message header related bug fixes
281
282 VIRGO code commits as on 25/6/2013
283 -----
284 1.telnet to kernel service was tested and found working
285 2.GFP_KERNEL changed to GFP_ATOMIC in VIRGO clouddriver kernel service
286
287 VIRGO code commits as on 1/7/2013
288 -----
289 1. Instead of printing iovect, printing buffer correctly prints the messages
290 2. wake_up_process() added and function received from virgo_clone() syscall is executed with kernel_thread and r
291 virgo_clone() syscall client.
292
293
294 commit as on 03/07/2013
295 -----
296 PRG loadbalancer preliminary code implemented. More work to be done
297
```



```
298 commit as on 10/07/2013
299 -----
300 Tested PRG loadbalancer read config code through telnet and virgo_clone. VFS code to read from virgo_cloud.conf
301
302 commits as on 12/07/2013
303 -----
304 PRG loadbalancer prototype has been completed and tested with test_virgo_clone and telnet and symbol export error
305
306 commits as on 16/07/2013
307 -----
308 read_virgo_config() and read_virgo_clone_config()(replica of read_virgo_config()) have been implemented and tested
309 all nodes). Thus minimal cloud functionality with config file support is in place. Todo things include function
310
311 commits as on 17/07/2013
312 -----
313 moved read_virgo_config() to VIRGOcloudexec's module_init so that config is read at boot time and exported symbols
314 Also commented read_virgo_clone_config() as it is redundant
315
316 commits as on 23/07/2013
317 -----
318
319 Lack of reflection kind of facilities requires map of function_names to pointers_to_functions to be executed
320 on cloud has to be lookedup in the map to get pointer to function. This map is not scalable if number of functions
321 in millions and size of the map increases linearly. Also having it in memory is both CPU and memory intensive.
322 Moreover this map has to be synchronized in all nodes for coherency and consistency which is another intensive task.
323 Thus name to pointer function table is at present not implemented. Suitable way to call a function by name of the function
324 is yet to be found out and references in this topic are scarce.
325
326 If parameterIsExecutable is set to 1 the data received from virgo_clone() is not a function but name of executable
327 This executable is then run on usermode using call_usermodehelper() which internally takes care of queueing the process
328 and executes the binary as child of keventd and reaps silently. Thus workqueue component of kernel is indirectly
329 This is sometimes more flexible alternative that executes a binary itself on cloud and
330 is preferable to clone()ing a function on cloud. Virgo_clone() syscall client or telnet needs to send the message
331
332 If parameterIsExecutable is set to 0 then data received from virgo_clone() is name of a function and is executed
333 using dlsym() lookup and pthread_create() in user space. This unifies both call_usermodehelper() and creating a process
334 with a fixed binary which is same for any function. The dlsym lookup requires mangled function names which need to be
335 virgo_clone or telnet. This is far more efficient than a function pointer table.
336
337 call_usermodehelper() Kernel upcall to usermode to exec a fixed binary that would inturn execute the cloneFunction
```

by spawning a pthread. cloneFunction is name of the function and not binary. This clone function will be dlsym() and a pthread will be created by the fixed binary. Name of the fixed binary is hardcoded herein as "virgo_kernelupcall_plugin". This fixed binary takes clone function as argument. For testing libvirgo.so has been virgo_cloud_test.c and separate build script to build the cloud function binaries has been added.

- Ka.Shrinivaasan (alias) Shrinivas Kannan (alias) Srinivasan Kannan
(<https://sites.google.com/site/kuja27>)

commits as on 24/07/2013

test_virgo_clone unit test case updated with mangled function name to be sent to remote cloud node. Tested with end-to-end and all features are working. But sometimes kernel_connect hangs randomly (this was observed only today to blocking vs non-blocking problem. Origin unknown).

- Ka.Shrinivaasan (alias) Shrinivas Kannan (alias) Srinivasan Kannan
(<https://sites.google.com/site/kuja27>)

commits as on 29/07/2013

Added kernel mode execution in the clone_func and created a sample kernel_thread for a cloud function. Some File binaries and parameterIsExecutable has been moved to virgo.h

commits as on 30/07/2013

New usecase virgo_cloud_test_kernelspace.ko kernel module has been added. This exports a function virgo_cloud_test accessed by virgo_cloudeexec kernel service to spawn a kernel thread that is executed in kernel addressspace. This on cloud adds a unique ability to VIRGO cloud platform to seamlessly integrate hardware devices on to cloud and to them from a remote cloud node through virgo_clone().

Thus above feature adds power to VIRGO cloud to make it act as a single "logical device driver" though devices a

commits as on 01/08/2013 and 02/08/2013

Added Bash shell commandline with -c option for call_usermodehelper upcall clauses to pass in remote virgo_clone arguments to it. Also tried output redirection but it works some times that too with a fatal kernel panic.

Ideal solutions are :

1. either to do a copy_from_user() for message buffer from user address space (or)

```
378 2. somehow rebuild the kernel with fd_install() pointing stdout to a VFS file* struct. In older kernels like 2.6
379 with in kmod.c (___call_usermodehelper()) which has been redesigned in kernel 3.x versions and fd_install has be
380 3. Create a Netlink socket listener in userspace and send message up from kernel Netlink socket.
381
382 All the above are quite intensive and time consuming to implement. Moreover doing FileIO in usermode helper is st
383
384 Since Objective of VIRGO is to virtualize the cloud as single execution "machine", doing an upcall (which would
385 redundant often and kernel mode execution is sufficient. Kernel mode execution with intermodule function invocat
386 the entire board in remote machine (since it can access PCI bus, RAM and all other device cards)
387
388 As a longterm design goal, VIRGO can be implemented as a separate protocol itself and sk_buff packet payload from
389 can be parsed by kernel service and kernel_thread can be created for the message.
390
391 commits as on 05/08/2013:
392 -----
393 Major commits done for kernel upcall usermode output logging with fd_install redirection to a VFS file. With thi
394
395 11 August 2013:
396 -----
397 Open Source Design and Academic Research Notes uploaded to http://sourceforge.net/projects/acadpdrafts/files/Mis
398
399
400 commits as on 23 August 2013
401 -----
402 New Multithreading Feature added for VIRGO Kernel Service - action item 5 in ToDo list above (virgo_cloudexec dr
403
404 commits as on 1 September 2013
405 -----
406 GNU Copyright license and Product Owner Profile (for identity of license issuer) have been committed. Also Virgo
407
408 commits as on 14 September 2013
409 -----
410 Updated virgo malloc design handwritten nodes on kmalloc() and malloc() usage in kernelspace and userspace execu
411
412 -----
413 VIRGO virtual addressing
414 -----
415 VIRGO virtual address is defined with the following datatype:
416
417 struct virgo_address
```

```

418 {
419     int node_id;
420     void* addr;
421 };
422
423 VIRGO address translation table is defined with following datatype:
424
425 struct virgo_addr_transtable
426 {
427     int node_id;
428     void* addr;
429 };
430
431 -----
432 VIRGO memory pooling prototypical implementation
433 -----
434 VIRGO memory pooling implementation as per the design notes committed as above is to be implemented as a prototy
435 under drivers/virgo/memorypooling and $LINUX_SRC_ROOT/virgo_malloc. But the underlying code is more or less simi
436
437 virgo_malloc() and related syscalls and virgo mempool driver connect to and listen on port different from cpupool
438
439 Commits as on 17 September 2013
440 -----
441 Initial untested prototype code - virgo_malloc and virgo mempool driver - for VIRGO Memory Pooling has been comm
442
443 Commits as on 19 September 2013
444 -----
445 3.7.8 Kernel full build done and compilation errors in VIRGO malloc and mempool driver code and more functions cr
446
447 Commits as on 23 September 2013
448 -----
449 Updated virgo_malloc.c with two functions, int_to_str() and addr_to_str(), using kmalloc() with full kernel re-bi
450 (Rather a re-re-build because some source file updates in previous build got deleted somehow mysteriously. This
451
452 Commits as on 24 September 2013
453 -----
454 Updated syscall*.tbl files, staging.sh, Makefiles for virgo_malloc(),virgo_set(),virgo_get() and virgo_free() mer
455
456 Commits as on 25 September 2013
457 -----

```

458 All build related errors fixed after kernel rebuild some changes made to function names to reflect their
459 names specific to memory pooling. Updated /etc/modules also has been committed to repository.
460
461 Commits as on 26 September 2013
462 -----
463 Circular dependency error in standalone build of cpu pooling and memory pooling drivers fixed and
464 datatypes and declarations for CPU pooling and Memory Pooling drivers have been segregated into respective header
465 virgo_mempool.h with corresponding service header files) to avoid any dependency error.
466
467 Commits as on 27 September 2013
468 -----
469 Major commits for Memory Pooling Driver listen port change and parsing VIRGO memory pooling commands have been done
470
471 Commits as on 30 September 2013
472 -----
473 New parser functions added for parameter parsing and initial testing on virgo_malloc() works with telnet client
474
475 Commits as on 1 October 2013
476 -----
477 Removed strcpy in virgo_malloc as ongoing bugfix for buffer truncation in syscall path.
478
479 Commits as on 7 October 2013
480 -----
481 Fixed the buffer truncation error from virgo_malloc syscall to mempool driver service which was caused by
482 sizeof() for a char*. BUF_SIZE is now used for size in both syscall client and mempool kernel service.
483
484 Commits as on 9 October 2013 and 10 October 2013
485 -----
486 Mempool driver kernelspace virgo mempool ops have been rewritten due to lack of facilities to return a
487 value from kernel thread function. Since mempool service already spawns a kthread, this seems to be sufficient. ,
488 causes the kernel socket to block as it waits for more data to be sent.
489
490 Commits as on 11 October 2013
491 -----
492 sscanf format error for virgo_cloud_malloc() return pointer address and sock_release() null pointer exception has
493 Added str_to_addr() utility function.
494
495 Commits as on 14 October 2013 and 15 October 2013
496 -----
497 Updated todo list

```
497 updated code list.
498
499 Rewritten virgo_cloud_malloc() syscall with:
500 - mutexed virgo_cloud_malloc() loop
501 - redefined virgo address translation table in virgo_mempool.h
502 - str_to_addr(): removed (void**) cast due to null sscanf though it should have worked
503
504 Commits as on 18 October 2013
505 -----
506 Continued debugging of null sscanf - added str_to_addr2() which uses simple_strtoll() kernel function
507 for scanning pointer as long long from string and casting it to void*. Also more %p qualifiers where
508 added in str_to_addr() for debugging.
509
510 Based on latest test_virgo_malloc run, simple_strtoll() correctly parses the address string into a long long base
511
512 Commits as on 21 October 2013
513 -----
514 Kern.log for testing after vtranstable addr fix with simple_strtoll() added to repository and still the other %p
515
516 Commits as on 24 October 2013
517 -----
518 Lot of bugfixes made to virgo_malloc.c for scanning address into VIRGO transtable and size computation. Testcase
519
520 Though the above sys_virgo_malloc() works, the return value is a kernel pointer if the virgo_malloc executes in
521
522 Commits as on 25 October 2013
523 -----
524 virgo_malloc.c has been rewritten by adding a userspace __user pointer to virgo_get() and virgo_set() syscalls with
525
526 Commits as on 29 October 2013
527 -----
528 Miscellaneous ongoing bugfixes for virgo_set() syscall error in copy_from_user().
529
530 Commits as on 2 November 2013
531 -----
532 Due to an issue which corrupts the kernel memory, presently telnet path to VIRGO mempool driver has been
533 tested after commits on 31 October 2013 and 1 November 2013 and is working but again there is an issue in kstrtou
534 data to set.
535
536 Commits as on 6 November 2013
537 -----
```

New parser function `virgo_parse_integer()` has been added to `virgo_cloud_mempool_kernelspace` driver module which is in `lib/kstrtox.c` and modified locally to add an if clause to discard quotes and unquotes. With this the telnet path and `virgo_set()` are working. Today's `kern.log` has been added to repository in `test_logs/`.

Commits as on 7 November 2013

In addition to `virgo_malloc` and `virgo_set`, `virgo_get` is also working through telnet path after today's commit for

Commits as on 11 November 2013

More testing done on telnet path for `virgo_malloc`, `virgo_set` and `virgo_get` commands which work correctly. But the `kmem_cache_trace_alloc` panics that follow each successful virgo command execution. `kern.log` for this has been added.

Commits as on 22 November 2013

More testing done on telnet path for `virgo_malloc`, `virgo_set` and `virgo_get` after commenting kernel socket shutdown mempool sendto code. Kernel panics do not occur after commenting kernel socket shutdown.

Commits as on 2 December 2013

Lots of testing were done on telnet path and syscall path connection to VIRGO mempool driver and screenshots for

Commits as on 5 December 2013

More testing on system call path done for `virgo_malloc()`, `virgo_set()` and `virgo_get()` system calls with test_virgo

VIRGO version 12.0 tagged.

Commits as on 12 March 2014

Initial VIRGO queueing driver implemented that flips between two internal queues: 1) a native queue implemented with structure `virgo_workqueue_request`.

Commits as on 20 March 2014

- VIRGO queue with additional boolean flags for its use as KingCobra queue
- KingCobra kernel space driver that is invoked by the VIRGO workqueue handler

Commits as on 30 March 2014

```
577 Commits as on 30 March 2014
578 -----
579 - VIRGO mempool driver has been augmented with use_as_kingcobra_service flags in CPU pooling and Memory pooling (
580
581 Commits as on 6 April 2014
582 -----
583 - VIRGO mempool driver recvfrom() function's if clause for KingCobra has been updated for REQUEST header format:
584
585 Commits as on 7 April 2014
586 -----
587 - generate_logical_timestamp() function has been implemented in VIRGO mempool driver that generates timestamps b
588
589 Commits as on 25 April 2014
590 -----
591 - client ip address in VIRGO mempool recvfrom KingCobra if clause is converted to host byte order from network b
592
593 Commits as on 5 May 2014
594 -----
595 - Telnet path commands for VIRGO cloud file system - virgo_cloud_open(), virgo_cloud_read(), virgo_cloud_write()
596
597 Commits as on 7 May 2014
598 -----
599 - Bugfixes to tokenization in kernel upcall plugin with strsep() for args passed on to the userspace
600
601 Commits as on 8 May 2014
602 -----
603 - Bugfixes to virgo_cloud_fs.c for kernel upcall (parameterIsExecutable=0) and with these the kernel to userspace
604
605 Commits as on 6 June 2014
606 -----
607 - VIRGO File System Calls Path implementation has been committed. Lots of Linux Full Build compilation errors fi
608
609 Commits as on 3 July 2014
610 -----
611 - More testing and bugfixes for VIRGO File System syscalls have been done. virgo_write() causes kernel panic.
612
613 7 July 2014 - virgo_write() kernel panic notes:
614 -----
615 warning within http://lxr.free-electrons.com/source/arch/x86/kernel/smp.c#L121:
616
617 static void native_smp_send_reschedule(int cpu)
```



```

617 static void native_smp_send_reschedule(int cpu,
618 {
619     if (unlikely(cpu_is_offline(cpu))) {
620         WARN_ON(1);
621         return;
622     }
623     apic->send_IPI_mask(cpumask_of(cpu), RESCHEDULE_VECTOR);
624 }

```

625 This is probably a fixed kernel bug in <3.7.8 but recurring in 3.7.8:

- 626 - <http://lkml.iu.edu/hypermail/linux/kernel/1205.3/00653.html>
- 627 - http://www.kernelhub.org/?p=3&msg=74473&body_id=72338
- 628 - <http://lists.openwall.net/linux-kernel/2012/09/07/22>
- 629 - https://bugzilla.kernel.org/show_bug.cgi?id=54331
- 630 - <https://bbs.archlinux.org/viewtopic.php?id=156276>
- 631
- 632
- 633

634 Commits as on 29 July 2014

635 -----

636 All VIRGO drivers(cloudfs, queuing, cpupooling and memorypooling) have been built on 3.15.5 kernel with some Make

637 -----

638 Commits as on 17 August 2014

639 -----

640 (FEATURE - DONE) VIRGO Kernel Modules and System Calls major rewrite for 3.15.5 kernel - 17 August 2014

641 -----

642 1. VIRGO config files have been split into /etc/virgo_client.conf and /etc/virgo_cloud.conf to delink the cloud
643 config parameters reading and to do away with oft occurring symbol lookup errors and multiple definition errors
644 node_ip_addrs_in_cloud - these errors are frequent in 3.15.5 kernel than 3.7.8 kernel.

645 2. Each VIRGO module and system call now reads the config file independent of others - there is a read_virgo_conf

646 3. New kernel module config has been added in drivers/virgo. This is for future prospective use as a config expo
647 be looked up by any other VIRGO module for config parameters.

648 4. include/linux/virgo_config.h has the declarations for all the config variables declared within each of the VII

649 5. Config variables in each driver and system call have been named with prefix and suffix to differentiate the m

650 6. In geographically distributed cloud virgo_client.conf has to be in client nodes and virgo_cloud.conf has to be

```
057
658 7. Above segregation largely simplifies the build process as each module and system call is independently built \
659
660 8. VIRGO File system driver and system calls have been tested with above changes and the virgo_open(),virgo_read
661
662 -----
663 Committed as on 23 August 2014
664 -----
665 Commenting use_as_kingcobra_service if clauses temporarily as disabling also doesnot work and only commenting the
666 works for VIRGO syscall path. Quite weird as to how this relates to the problem. As this is a heisenbug further
667 difficult and sufficient testing has been done with logs committed to repository. Probably a runtime symbol looku
668 causes the freeze.
669 For forwarding messages to KingCobra and VIRGO queues, cpupooling driver is sufficient which also has the use_as_
670
671 -----
672 Committed as on 23 August 2014 and 24 August 2014
673 -----
674 As cpupooling driver has the same crash problem with kernel_accept() when KingCobra has benn enabled, KingCobra \
675
676 VIRGO cpupooling or memorypooling ==> VIRGO Queue =====> KingCobra
677
678 (or)
679 VIRGO Queue kernel service =====> KingCobra
680
681 -----
682 Committed as on 26 August 2014
683 -----
684 - all kmallocs have been made into GFP_ATOMIC instead of GFP_KERNEL
685 - moved some kingcobra related header code before kernel_recvmsg()
686 - some header file changes for set_fs()
687
688 This code has been tested with modified code for KingCobra and the standalone
689 kernel service that accepts requests from telnet directly at port 60000, pushes to virgo_queue
690 and is handled to invoke KingCobra servicerequest kernelspace function, works
691 (the kernel_recvmsg() crash was most probably due to Read-Only filesystem -errno printed is -30)
692
693 -----
694 VIRGO version 14.9.9 has been release tagged on 9 September 2014
695 -----
696
697 -----
```

```

697 -----
698 Committed as on 26 November 2014
699 -----
700 New kernel module cloudsync has been added to repository under drivers/virgo that can be used for synchronization
701 -----
702 -----
703 Committed as on 27 November 2014
704 -----
705 virgo_bakery.h bakery_lock() has been modified to take 2 parameters - thread_id and number of for loops (1 or 2)
706 -----
707 -----
708 Committed as on 2 December 2014
709 -----
710 VIRGO bakery algorithm implementation has been rewritten with some bugfixes. Sometimes there are soft lockup errors
711 -----
712 -----
713 Committed as on 17 December 2014
714 -----
715 Initial code commits for VIRGO EventNet kernel module service:
716 -----
717 1.EventNet Kernel Service listens on port 20000
718
719 2.It receives eventnet log messages from VIRGO cloud nodes and writes the log messages
720 after parsing into two text files /var/log/eventnet/EventNetEdges.txt and
721 /var/log/eventnet/EventNetVertices.txt by VFS calls
722
723 3.These text files can then be processed by the EventNet implementations in AsFer (python pygraph and
724 C++ boost::graph based)
725
726 4.Two new directories virgo/utils and virgo/eventnet have been added.
727
728 5.virgo/eventnet has the new VIRGO EventNet kernel module service implementation that listens on
729 port 20000.
730
731 6.virgo/utils is the new generic utilities driver that has a virgo_eventnet_log()
732 exported function which connects to EventNet kernel service and sends the vertex and edge eventnet
733 log messages which are parsed by kernel service and written to the two text files above.
734
735 7.EventNet log messages have two formats:
736   - Edge message - "eventnet_edgemsg#<id>#<from_event>#<to_event>"
737   - Vertex message - "eventnet_vertexmsg#<id>#<partakers_csv>#<partaker_conversations_csv>"

```

```

- vertex message - eventnet_vertexmsg.h - parameters csv - parameters conversations csv

```

8.The utilities driver Module.symvers have to be copied to any driver which are then merged with the symbol files of the corresponding driver. Target clean has to be commented while building the unified Module.symvers because it erases symvers carried over earlier.

9.virgo/utls driver can be populated with all necessary utility exported functions that might be needed in other VIRGO drivers.

10.Calls to virgo_eventnet_log() have to be #ifdef guarded as this is quite network intensive.

Commits as on 18 December 2014

Miscellaneous bugfixes,logs and screenshot

- virgo_clouddexec_eventnet.c - eventnet messages parser errors and eventnet_func bugs fixed
- virgo_cloud_eventnet_kernelspace.c - filp_open() args updated due to vfs_write() kernel panics. The vertexmess:
- VIRGO EventNet build script updated for copying Module.symvers from utls driver for merging with eventnet Mod
- Other build generated sources and kernel objects
- new testlogs directory with screenshot for edgmsg sent to EventNet kernel service and kern.log with previous l
- vertex message update

Commits as on 2,3,4 January 2015

- fixes for virgo eventnet vertex and edge message text file vfs_write() errors
- kern.logs and screenshots

VIRGO version 15.1.8 release tagged on 8 January 2015

Commits as on 3 March 2015 - Initial commits for Kernel Analytics Module which reads the /etc/virgo_kernel_analy:

- Architecture of Key-Value Store in memorypooling (virgo_malloc,virgo_get,virgo_set,virgo_free) has been uploaded as a diagram at http://sourceforge.net/p/virgo-linux/code-0/HEAD/tree/trunk/virgo-docs/VIRGOLinuxKernel_
- new kernel_analytics driver for AsFer <=> VIRGO+USBmd+KingCobra interface has been added.
- virgo_kernel_analytics.conf having csv(s) of key-value pairs of analytics variables is set by AsFer or any oth

```

777 - virgo_kernel_analytics.com having csv(s) of key-value pairs of analytics variables is set by ASSET or any other
778 - kernel_analytics Driver build script has been added
779
780 -----
781 Commits as on 6 March 2015
782 -----
783 - code has been added in VIRGO config module to import EXPORTed kernel_analytics config key-pair array
784 set by Apache Spark (mined from Uncomplicated Fire Wall logs) and manually and write to kern.log.
785
786 -----
787 NeuronRain version 15.6.15 release tagged
788 -----
789
790 -----
791 Portability to linux kernel 4.0.5
792 -----
793 The VIRGO kernel module drivers are based on kernel 3.15.5. With kernel 4.0.5 kernel which is the latest followin
794 compilation and LD errors occur - this is on cloudfs VIRGO File System driver :
795 - msghdr has to be user_msghdr for iov and iov_len as there is a segregation of msghdr
796 - modules_install throws an error in scripts/Makefile.modinst while overwriting already installed module
797
798 -----
799 Commits as on 9 July 2015
800 -----
801 VIRGO cpupooling driver has been ported to linux kernel 4.0.5 with msghdr changes as mentioned previously
802 with kern.log for VIRGO cpupooling driver invoked in parameterIsExecutable=2 (kernel module invocation)
803 added in testlogs
804
805 -----
806 Commits as on 10,11 July 2015
807 -----
808 VIRGO Kernel Modules:
809 - mempooling
810 - cloudfs
811 - utils
812 - config
813 - kernel_analytics
814 - cloudsync
815 - eventnet
816 - queuing
817 along with cpupooling have been ported to Linux Kernel 4.0.5 - Makefile and header files have been

```

along with epapouting have been ported to Linux Kernel 4.0.5 - Makefile and header files have been updated wherever required.

Commits as on 20,21,22 July 2015

Due to SourceForge Storage Disaster(<http://sourceforge.net/blog/sourceforge-infrastructure-and-service-restoration>) the github replica of VIRGO is urgently updated with some important changes for msg_iter, iovec etc., in 4.0.5 kernel port specifically for KingCobra and VIRGO Queueing. These have to be committed to SourceForge repository at http://sourceforge.net/users/ka_shrinivaasan once SourceForge repos are restored. Time to move on to the manufacturing hub? GitHub ;-)

VIRGO Queueing Kernel Module Linux Kernel 4.0.5 port:

- msg_iter is used instead of user_msghdr
- kvec changed to iovec
- Miscellaneous BUF_SIZE related changes
- kern.logs for these have been added to testlogs
- Module.symvers has been recreated with KingCobra Module.symvers from 4.0.5 KingCobra build
- clean target commented in build script as it wipes out Module.symvers
- updated .ko and .mod.c

KingCobra Module Linux Kernel 4.0.5 port

- vfs_write() has a problem in 4.0.5
- the filp_open() args and flags which were working in 3.15.5 cause a kernel panic implicitly and nothing was written to logs
- It took a very long time to figure out the reason to be vfs_write and filp_open
- O_CREAT, O_RDWR and O_LARGEFILE cause the panic and only O_APPEND is working, but does not do vfs_write(). All other VIRGO Queue + KingCobra functionalities work viz., enqueueing, workqueue handler invocation, dequeueing, invoking kingcobra kernelspace service request function from VIRGO queue handler, timestamp, timestamp and IP parser, reply_to_publisher etc.,
- As mentioned in Greg Kroah Hartman's "Driving me nuts", persistence in Kernel space is a bad idea but still seems to be a necessary stuff - yet only vfs calls are used which have to be safe
- Thus KingCobra has to be in-memory only in 4.0.5 if vfs_write() doesn't work
- Intriguingly cloudfs filesystems primitives - virgo_cloud_open, virgo_cloud_read, virgo_cloud_write etc., work perfectly and append to a file.
- kern.logs for these have been added to testlogs
- Module.symvers has been recreated for 4.0.5
- updated .ko and .mod.c

```

857 -----
858
859 Due to SourceForge outage and for a future code diversification
860 NeuronRain codebases (AsFer, USBmd, VIRGO, KingCobra)
861 in http://sourceforge.net/u/userid-769929/profile/ have been
862 replicated in GitHub also - https://github.com/shrinivaasanka
863 excluding some huge logs due to Large File Errors in GitHub.
864 -----
865
866 -----
867 Commits as on 30 July 2015
868 -----
869 VIRGO system calls have been ported to Linux Kernel 4.0.5 with commented gcc option -Wimplicit-function-declarat:
870 msghdr and iovec changes similar to drivers mentioned in previous commit notes above. But Kernel 4.1.3 has some I
871 The NeuronRain codebases in SourceForge and GitHub would henceforth be mostly and always out-of-sync and not gua
872
873 -----
874 Commits as on 2,3 August 2015
875 -----
876 - new .config file added which is created from menuconfig
877 - drivers/Kconfig has been updated with 4.0.5 drivers/Kconfig for trace event linker errors
878 Linux Kernel 4.0.5 - KConfig is drivers/ has been updated to resolve RAS driver trace event linker error. RAS wa:
879 - link-vmlinux.sh has been replaced with 4.0.5 kernel version
880
881 -----
882 Commits as on 12 August 2015
883 -----
884 VIRGO Linux Kernel 4.1.5 port - related code changes - some important notes:
885 -----
886 - Linux Kernel 4.0.5 build suddenly had a serious root shell drop error in initramfs which was not resolved by:
887   - adding rootdelay in grub
888   - disabling uuid for block devices in grub config
889   - mounting in read/write mode in recovery mode
890   - no /dev/mapper related errors
891   - repeated exits in root shell
892   - delay before mount of root device in initrd scripts
893 - mysteriously there were some firmware microcode bundle executions in ieucodetool
894 - Above showed a serious grub corruption or /boot MBR bug or 4.0.5 VIRGO kernel build problem
895 - Linux 4.0.x kernels are EOL-ed
896 - Hence VIRGO is ported to 4.1.5 kernel released few days ago
897 - Only minimum files have been changed as in commit log for Makefiles and syscall table and headers and a build

```

```
897 - Only minimum files have been changed as in commit log for Makefiles and syscall table and headers and a build script
898 for 4.1.5:
899     Changed paths:
900     A buildscript_4.1.5.sh
901     M linux-kernel-extensions/Makefile
902     M linux-kernel-extensions/arch/x86/syscalls/Makefile
903     M linux-kernel-extensions/arch/x86/syscalls/syscall_32.tbl
904     M linux-kernel-extensions/drivers/Makefile
905     M linux-kernel-extensions/include/linux/syscalls.h
906
907 - Above minimum changes were enough to build an overlay-ed Linux Kernel with VIRGO codebase
908
909 -----
910 Commits as on 14,15,16 August 2015
911 -----
912 Executed the minimum end-end telnet path primitives in Linux kernel 4.1.5 VIRGO code:
913 - cpu virtualization
914 - memory virtualization
915 - filesystem virtualization (updated filp_open flags)
916 and committed logs and screenshots for the above.
917
918 -----
919 Commits as on 17 August 2015
920 -----
921 VIRGO queue driver:
922 - Rebuilt Module.symvers
923 - kern.log for telnet request to VIRGO Queue + KingCobra queueing system in kernelspace
924
925 -----
926 Commits as on 25,26 September 2015
927 -----
928 VIRGO Linux Kernel 4.1.5 - memory system calls:
929 -----
930 - updated testcases and added logs for syscalls invoked separately(malloc,set,get,free)
931 - The often observed unpredictable heisen kernel panics occur with 4.1.5 kernel too. The logs are 2.3G and
932 only grepped output is committed to repository.
933 - virgo_malloc.c has been updated with kstrdup() to copy the buf to iov.iov_base which was earlier
934 crashing in copy_from_iter() within tcp code. This problem did not happen in 3.15.5 kernel.
935 - But virgo_clone syscall code works without any changes to iov_base as above which does a strcpy()
936 which is an internal memcpy() though. So what causes this crash in memory system calls alone
937 is a mystery
```



```
937  is a mystery.
938  - new insmod script has been added to load the VIRGO memory modules as necessary instead of at boot time.
939  - test_virgo_malloc.c and its Makefile has been updated.
940
941  VIRGO Linux Kernel 4.1.5 - filesystem calls- testcases and logs:
942  -----
943  - added insmod script for VIRGO filesystem drivers
944  - test_virgo_filesystem.c has been updated for syscall numbers in 4.1.5 VIRGO kernel
945  - virgo_fs.c syscalls code has been updated for iov.iov_base kstrdup() - without this there are kernel panics :
946  testlogs have been added, but there are heisen kernel panics. The virgo syscalls are executed but not written to
947  Thus execution logs are missing for VIRGO filesystem syscalls.
948
949  -----
950  Commits as on 28,29 September 2015
951  -----
952
953  VIRGO Linux Kernel 4.1.5 filesystem syscalls:
954  -----
955  - Rewrote iov_base code with a separate iovbuf set to iov_base and strcpy()-ing the syscall command to iov_base :
956  memory syscalls
957  - Pleasantly the same iovbuf code that crashes in memory syscalls works for VIRGO FS without crash. Thus both virgo
958  syscalls work without issues in 4.1.5 and virgo_malloc() works erratically in 4.1.5 which remains as issue.
959  - kern.log for VIRGO FS syscalls and virgofstest text file written by virgo_write() have been added to repository
960
961
962  VIRGO Linux 4.1.5 kernel memory syscalls:
963  -----
964  - rewrote the iov_base buffer code for all VIRGO memory syscalls by allocating separate iovbuf and copying the m
965  - did extensive repetitive tests that were frequented by numerous kernel panics and crashes
966  - The stability of syscalls code with 3.15.5 kernel appears to be completely absent in 4.1.5
967  - The telnet path works relatively better though
968  - Difference between virgo_clone and virgo_malloc syscalls despite having same kernel sockets code looks like a
969  - kernel OOPS traces are quite erratic.
970  - Makefile path in testcase has been updated
971
972  -----
973  Commits as on 4 October 2015
974  -----
975  VIRGO Linux Kernel 4.1.5 - Memory System Calls:
976  -----
```

```

977 - replaced copy_to_user() with a memcpy()
978 - updated the testcase with an example VUID hardcoded.
979 - str_to_addr2() is done on iov_base instead of buf which was causing NULL parsing
980 - kern.log with above resolutions and multiple VIRGO memory syscalls tests - malloc,get,set
981 - With above VIRGO malloc and set syscalls work relatively causing less number of random kernel panics
982 - return values of memory calls set to 0
983 - in virgo_get() syscall, memcpy() of iov_base is done to data_out userspace pointer
984 - kern.log with working logs for syscalls - virgo_malloc(), virgo_set(), virgo_get() but still there are random I
985 - Abridged kern.log for VIRGO Memory System Calls with 4.1.5 Kernel - shows example logs for virgo_malloc(), virg
986
987 -----
988 Commits as on 14 October 2015
989 -----
990 VIRGO Queue Workqueue handler usermode clause has been updated with 4.1.5 kernel paths and kingcobra in user mode
991
992 -----
993 Commits as on 15 October 2015
994 -----
995 - Updated VIRGO Queue kernel binaries and build generated sources
996 - virgo_queue.h has been modified for call_usermodehelper() - set_ds() and fd_install() have been uncommented fo
997
998 -----
999 Commits as on 3 November 2015
1000 -----
1001 - kern.log for VIRGO kernel_analytics+config drivers which export the analytics variables from /etc/virgo_kernel_
1002
1003 -----
1004 Commits as on 10 January 2016
1005 -----
1006 NeuronRain VIRGO Research version 2016.1.10 released.
1007
1008 -----
1009 NeuronRain - AsFer commits for VIRGO - C++ and C Python extensions
1010 - Commits as on 29 January 2016
1011 -----
1012 -----
1013 (FEATURE - DONE) Python-C++-VIRGOKernel and Python-C-VIRGOKernel boost::python and cpython implementations:
1014 -----
1015 - It is a known idiom that Linux Kernel and C++ are not compatible.
1016 - In this commit an important feature to invoke VIRGO Linux Kernel from userspace python libraries via two alteri

```

```

1017 - In one alternative, C++ boost::python extensions have been added to encapsulate access to VIRGO memory system
1018 - In the other alternative, C Python extensions have been added that replicate boost::python extensions above in
1019 works exceedingly well compared to boost::python.
1020 - This functionality is required when there is a need to set kernel analytics configuration variables learnt by ,
1021 dynamically without re-reading /etc/virgo_kernel_analytics.conf.
1022 - This completes a major integration step of NeuronRain suite - request travel roundtrip to-and-fro top level ma
1023 code and rock-bottom C linux kernel - bull tamed ;-).
1024 - This kind of python access to device drivers is available for Graphics Drivers already on linux (GPIO - for ac
1025 - logs for both C++ and C paths have been added in cpp_boost_python_extensions/ and cpython_extensions.
1026 - top level python scripts to access VIRGO kernel system calls have been added in both directories:
1027     CPython - python cpython_extensions/asferpythonextensions.py
1028     C++ Boost::Python - python cpp_boost_python_extensions/asferpythonextensions.py
1029 - .so, .o files with build commandlines(asferpythonextensions.build.out) for "python setup.py build" have been a
1030 in build lib and temp directories.
1031 - main implementations for C++ and C are in cpp_boost_python_extensions/asferpythonextensions.cpp and cpython_ex
1032

```

```

1033 -----
1034 Commits as on 12 February 2016
1035 -----

```

```

1036 Commits for Telnet/System Call Interface to VIRGO CPUPooling -> VIRGO Queue -> KingCobra
1037 -----

```

```

1038 *) This was commented earlier for the past few years due to a serious kernel panic in previous kernel versions -
1039 *) In 4.1.5 a deadlock between VIRGO CPUPooling and VIRGO queue driver init was causing following error in "use_
1040     - "gave up waiting for virgo_queue init, unknown symbol push_request()"
1041 *) To address this a new boolean flag to selectively enable and disable VIRGO Queue kernel service mode "virgo_q
1042 *) With this flag VIRGO Queue is both a kernel service driver and a standalone exporter of function symbols - pu
1043 *) Incoming request data from telnet/virgo_clone() system call into cpupooling kernel service reactor pattern (v
1044 *) This resolves a long standing deadlock above between VIRGO cpupooling "use_as_kingcobra_service" clause and V
1045 *) This makes virgo_clone() systemcall/telnet both synchronous and asynchronous - requests from telnet client/vi
1046 *) Above saves an additional code implementation for virgo_queue syscall paths - virgo_clone() handles, based on
1047 -----

```

```

1048 Prerequisites:
1049 -----

```

```

1050 - insmod kingcobra_main_kernelspace.ko
1051 - insmod virgo_queue.ko compiled with flag virgo_queue_reactor_service_mode=1
1052     (when virgo_queue_reactor_service_mode=0, listens on port 60000 for direct telnet requests)
1053 - insmod virgo_cloud_test_kernelspace.ko
1054 - insmod virgo_clouddexec.ko (listens on port 10000)
1055 -----
1056

```

```
1057 Schematic Diagram
1058 -----
1059 VIRGO clone system call/telnet client ---> VIRGO cpupooling(compiled with use_as_kingcobra_service=1) -----> VII
1060
1061 -----
1062 Commits as on 15 February 2016 - Kernel Analytics - VIRGO Linux Kernelwide imports
1063 -----
1064 - Imported Kernel Analytics variables into CloudFS kernel module - printed in driver init()
1065 - Module.symvers from kernel_analytics has been merged with CloudFS Module.symvers
1066 - Logs for above has been added in cloudfs/test_logs/
1067 - Makefile updated with correct fs path
1068 - Copyleft notices updated
1069
1070 -----
1071 Commits as on 15 February 2016 - Kernel Analytics - VIRGO Linux Kernelwide imports
1072 -----
1073 - Kernel Analytics driver exported variables have been imported in CPU virtualization driver
1074 - Module.symvers from kernel_analytics has been merged with Module.symvers in cpupooling
1075 - kern.log for this import added to cpupooling/virgocloudexec/test_logs/
1076
1077 -----
1078 Commits as on 15 February 2016 - Kernel Analytics - VIRGO Linux Kernelwide imports
1079 -----
1080 - Imported kernel analytics variables into memory virtualization driver init() , exported from kernel_analytics (
1081 - build shell script updated
1082 - logs added to test_logs/
1083 - Module.symvers from kernel_analytics has been merged with memory driver Module.symvers
1084 - Makefile updated
1085
1086 -----
1087 Commits as on 15 February 2016 - Kernel Analytics - VIRGO Linux Kernelwide imports
1088 -----
1089 - Imported kernel analytics variables into VIRGO Queueing Driver
1090 - logs for this added in test_logs/
1091 - Makefile updated
1092 - Module.symvers from kernel_analytics has been merged with Queueing driver's Module.symvers
1093 - .ko, .o and build generated sources
1094
1095 -----
1096 Commits as on 16,17 February 2016
```

(FEATURE-DONE) Socket Buffer Debug Utility Function - uses linux skbuff facility

- In this commit a multipurpose socket buffer debug utility function has been added in utils driver and exported
- It takes a socket as function argument does the following:
 - dereference the socket buffer head of skbuff per-socket transmit data queue
 - allocate skbuff with alloc_skb()
 - reserve head room with skb_reserve()
 - get a pointer to data payload with skb_put()
 - memcpy() an example const char* to skbuff data
 - Iterate through the linked list of skbuff queue in socket and print headroom and data pointers
 - This can be used as a packet sniffer anywhere within VIRGO linux network stack
- Any skb_*() functions can be plugged-in here as deemed necessary.
- kern.log(s) which print the socket internal skbuff data have been added to a new testlogs/ directory
- .cmd files generated by kbuild

(FEATURE-DONE) Commits as on 24 February 2016

skbuff debug function in utils/ driver:

- (*) Added an if clause to check NULLity of skbuff headroom before doing skb_alloc()
- (*) kern.log for this commit has been added testlogs/
- (*) Rebuilt kernel objects and sources

Commits as on 1 March 2016

(FEATURE-DONE) Software Analytics - SATURN Program Analysis added to VIRGO Linux kernel drivers

- SATURN (saturn.stanford.edu) Program Analysis and Verification software has been integrated into VIRGO Kernel as a Verification+SoftwareAnalytics subsystem
- A sample driver that can invoke an exported function has been added in drivers - saturn_program_analysis
- Detailed document for an example null pointer analysis usecase has been created in virgo-docs/VIRGO_SATURN_Proj
- linux-kernel-extensions/drivers/virgo/saturn_program_analysis/saturn_program_analysis_trees/error.txt is the e
- SATURN generated preproc and trees are in linux-kernel-extensions/drivers/virgo/saturn_program_analysis/prepro
- linux-kernel-extensions/drivers/virgo/saturn_program_analysis/saturn_program_analysis_trees/

Commits as on 10 March 2016

```
-----
1137 - SATURN analysis databases (.db) for locking, memory and CFG analysis.
1138 - DOT and PNG files for locking, memory and CFG analysis.
1139 - new folder saturn_calypso_files/ has been added in saturn_program_analysis/ with new .clp files virgosaturncfg
1140 - SATURN alias analysis .db files
1141
1142 -----
1143 (FEATURE-DONE) NEURONRAIN - ASFER Commits for VIRGO - CloudFS systems calls integrated into Boost::Python C++ and
1144 -----
1145 -----
1146 AsFer Commits as on 30 May 2016
1147 -----
1148 VIRGO CloudFS system calls have been added (invoked by unique number from syscall_32.tbl) for C++ Boost::Python :
1149 VIRGO Linux System Calls. Switch clause with a boolean flag has been introduced to select either VIRGO memory or
1150 kern.log and CloudFS textfile Logs for VIRGO memory and filesystem invocations from AsFer python have been commi
1151 -----
1152 AsFer Commits as on 31 May 2016
1153 -----
1154 Python CAPI interface to NEURONRAIN VIRGO Linux System Calls has been updated to include File System open, read,
1155 Rebuilt extension binaries, kern.logs and example appended text file have been committed to testlogs/. This is e
1156 commits done for Boost::Python C++ interface. Switch clause has been added to select memory or filesystem VIRGO :
1157 -----
1158 (BUG - STABILITY ISSUES) Commits - 25 July 2016 - Static Analysis of VIRGO Linux kernel for investigating heisen
1159 -----
1160 Initial Documentation for Smatch and Coccinelle kernel static analyzers executed on VIRGO Linux kernel - to be up
1161 periodically with further analysis.
1162 -----
1163 (BUG - STABILITY ISSUES) Commits - 1 August 2016 - VIRGO Linux Stability Issues - Ongoing Random oops and panics
1164 -----
1165 1. GFP_KERNEL has been replaced with GFP_ATOMIC flags in kmem allocations.
1166 2. NULL checks have been introduced in lot of places involving strcpy, strcat, strcmp etc., to circumvent
1167 buffer overflows.
1168 3. Though this has stabilized the driver to some extent, still there are OOPS in unrelated places deep
1169 with in kernel where paging datastructures are accessed - kmalloc somehow corrupts paging
1170 4. OOPS are debugged via gdb as:
1171     4.1 gdb ./vmlinux /proc/kcore
1172     or
1173
```

```
1177         4.2 gdb <loadable_kernel_module>.o
1178         followed by
1179         4.3 l *(address+offset in OOPS dump)
1180 5. kern.log(s) for the above have been committed in tar.gz format and have numerous OOPS occurred during repetit:
1181 invocation(boost::python C++) invocations of virgo memory system calls.
1182 6. Paging related OOPS look like an offshoot of set_fs() encompassing the filp_open VFS calls.
1183
1184 -----
1185 (BUG-STABILITY ISSUES) Commits - 26 September 2016 - Ongoing Random Panic investigation
1186 -----
1187 Further analysis on direct VIRGO memory cache primitives telnet invocation - problems are similar
1188 to Boost::Python AsFer VIRGO system calls invocations.
1189
1190 -----
1191 (BUG-STABILITY ISSUES) Commits - 27 September 2016 - Ongoing Random Panic investigation
1192 -----
1193 Analysis of VIRGO memory cache primitives reveal more inconsistencies in cacheline flushes between CPU and GPU.
1194
1195
1196
1197 Srinivasan Kannan (alias) Ka.Shrinivaasan (alias) Shrinivas Kannan
1198 http://sites.google.com/site/kuja27
```