KingCobra (/p/kcobra/)

Brought to you by: ka_shrinivaasan (/u/userid-769929/)

[r168] (/p/kcobra/code-svn/168/): (./) / KingCobraDesignNotes.txt Readme (#) History (/p/kcobra/code-svn/168/log/?path=/KingCobraDesignNotes.txt)

Download this file (?format=raw)

409 lines (313 with data), 48.3 kB

```
-------------------------------------------------------------------------------------------
KingCobra - A Research Software for Distributed Request Service on Cloud with Arbiters

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.

#--------------------------------------------------------------------------------------------------------
#Copyleft (Copyright+):
#Srinivasan Kannan
#(also known as: Ka.Shrinivaasan, Shrinivas Kannan)
#Ph: 9791499106, 9003082186
#Krishna iResearch Open Source Products Profiles:
#http://sourceforge.net/users/ka_shrinivaasan,
#https://github.com/shrinivaasanka,
#https://www.openhub.net/accounts/ka_shrinivaasan
#Personal website(research): https://sites.google.com/site/kuja27/
#emails: ka.shrinivaasan@gmail.com, shrinivas.kannan@gmail.com,
#kashrinivaasan@live.com
#--------------------------------------------------------------------------------------------------------
*****************************************************************************************/

Theoretical Interludes, Design Notes and ToDo (long term with no deadline)
-------------------------------------------------------------------------

[This is a major research oriented subsystem of NeuronRain and inspired by COBRA project (done by the author in his BE

[KingCobra though a misnomer is expanded as ClOud With ARBiters MimicKING containing the anagram]

1. (THEORY) There is a cloud of nodes which execute a set of services from randomly created clients.

2. (THEORY) This cloud could be on iCloud (AsFer+USBmd+VIRGO) platform or any other opensource cloud platforms like Had

3. (THEORY) The Clients are publishers of Service requests which are of many types -  miscellaneous types of Service th

4. (THEORY) The Services on the Cloud are Subscribers to these requests of specific type. Thus this is the conventional

5. (THEORY) The requests flow through cloud using a workqueue (which could be a lowlevel Linux workqueue or VIRGO queue

6. (THEORY) The difference is that the Cloud has nodes that "deceive" or "corrupt".

7. (THEORY) Service requests - are published by the clients in the need of a service which could be defined by markup f

8. (THEORY) Problem reports - are published by clients which are "dissatisfied" by the quality of service by the cloud.

9. (THEORY) Suggestions - are enhancement requests sent by clients and require manual intervention.

10. (THEORY) Cloud nodes have a Quality of Service metric calculated by a model.
```

11. (THEORY) The cloud has a reporting structure of nodes - either as a graph or tree. The graph is dynamically reorgan

12. (THEORY) The difficult part of the above is using Arbiters to find "faulty" nodes based on problem reports from cli

13. (THEORY) Brewer's CAP conjecture proved by [GilbertLynch] as a theorem (still debated) states that only 2 of the 3

14. (THEORY) CAP theorem does not seem to apply to the above faulty scenario with corrupt nodes under Consistency or Av

15. (THEORY) As "corruption" is more conspicuous with monetary element, if above services are "charged" with a logical

16. (THEORY) Identifying criminal nodes as in (15) above seems to be beyond the ambit of CAP. Thus CAP with Integrity f

17. (THEORY-ONGOING) Analytics on the Problem reports sent to the cloud queue give a pattern of corrupt nodes. Intrinsi

18. (THEORY) Policing the cloud nodes with arbiters - This seems to be limited by CAP theorem and Integrity as above. A

19. (THEORY) Brooks-Iyengar algorithm for sensors in all cloud nodes is an improved Byazantine Fault Tolerant algorithm

20. (THEORY) BitCoin is a Byzantine Fault Tolerant protocol.

21. (THEORY) Byzantine Fault Tolerance in Clouds is described in http:*//www.computer.org/csdl/proceedings/cloud/2011/44*

22. (THEORY) Related to point 18 - The problem of fact finding or fault finding using a cloud police has the same limit

23. (THEORY) Reference article on cloud BFT for Byzantine, Corrupt brokers - Byzantine Fault-Tolerant Publish/Subscribe

```
---------------------------------------------------------------
24. KingCobra messaging request-response design - options
---------------------------------------------------------------
```

24a. Implementing a message subscription model in kernelspace where clients publish the message that is queued-in to su

24b. (DONE-minimal implementation) At present a minimum kernelspace messaging system that queues remote request and han

```
-----------------------------------------------------------------------------------------------------
24c.(DONE) KingCobra - VIRGO queue - VIRGO cpupooling , mempooling and queue service drivers interaction schematic diag
-----------------------------------------------------------------------------------------------------

        KingCobraClient ==========>=<REQUEST:id>===================> VIRGO cpupooling service =====> VIRGO Queue ======
               ||
               ||
                <================= VIRGO Queue <====== VIRGO cpupooling service ====<REPLY:id>==========================

        KingCobraClient ==========>=<REQUEST:id>===================> VIRGO mempooling service =====> VIRGO Queue ======
               ||
               ||
                <================= VIRGO Queue <====== VIRGO mempooling service ====<REPLY:id>==========================

        KingCobraClient ==========>=<REQUEST:id>===================> VIRGO Queue service ==============================
               ||
               ||
                <================= VIRGO Queue service ============================<REPLY:id>==========================
```

24d. (ONGOING) kingcobra_servicerequest_kernelspace() distinguishes the "REQUEST" and "REPLY" and optionally persists t

24e.Above option 24b implements a simple p2p queue messaging in kernel. To get a Topic-like behaviour in VIRGO queue mi

25. (ONGOING) For the timestamp service, EventNet described in http:*//sourceforge.net/p/asfer/code/HEAD/tree/AstroInfer*

26. (THEORY - ONGOING Implementation) MESSAGE-AS-CURRENCY PROTOCOL: If each message payload is also construed as a curr

```
        m1=MAC_alloc(denomination)
        m2=m1 (---- this is disallowed)
```

Linux kernel allocation functions - kmalloc() - have a krefs functionality for reference counting within kernel. Refcou

```
26.1 Schematic Diagram for Cloud Perfect Forwarding with AsFer+VIRGOQueue+KingCobraUserspace:
---------------------------------------------------------------------------------------------

    Telnet or other client ===========> VIRGO Queue Service Listener ======> VIRGO Workqueue Handler
                                                                                |
                                                                                |  [KernelSpace]
                                                                                V
        AsFer Cloud Perfect Forwarding Client <===== KingCobra Userspace shell script(call_usermodehelper)
                        |                                                       |
                        |                                                       |
  Virtual Currency      |                                                       |  [UserSpace]
                        V                                                       V
        AsFer Cloud Perfect Forwarding Server <=================================/
```

```


References:
----------
26.2 An example distributed transactional memory implementation in cloud - http://infinispan.org/tutorials/simple/tx/ a

27. (THEORY) SIMULATING A VIRTUAL ECONOMY with above MAC protocol (Message-as-currency): If each message sent is conside

28. (THEORY) TRADING WITH ABOVE KINGCOBRA MAC protocol - somewhat oversimplified:

                                --------------------
                                |Unique MAC id MINT|
                                --------------------
                                        ||
        -----money trail------------------
        |
        V
        ....
        Buyer   ======= sends MAC message (REQUEST id) =======> Seller (stores the MAC in local cash reserve and prepen
        ||                                                     ||
        <============ sends the goods and services (REPLY id) ===

In the above schematic, money with unique id in cloud reaches a buyer after many buyer-seller transitions called "money

References:
-----------
28.1 Price fixing for items in Buyer-Seller-Trader networks - Trading Networks - Market Equilibrium and Walrasian Model
28.2 Algoithmic Game Theory - Market Equilibrium for Price - Equilibrium is a strategic standoff - both players can't b
28.3 Price-setting in Trading networks - Chapter 11 - https://www.cs.cornell.edu/home/kleinber/networks-book/networks-b

29. (THEORY) VALUE FOR ELECTRONIC MONEY: How is above MAC money earned - This again requires linking value to money (as

30. (THEORY and IMPLEMENTATION) FIXING VALUE FOR MAC MONEY: To delineate corruption as discussed in 27 above with value
        --------------------------------------------------------------------------------------------------
        |value(i) = summation(value(ingredients of i)) + cost(integrating the ingredients to create item i) |
        --------------------------------------------------------------------------------------------------
Obviously the above recursion combinatorially explodes into exponential number of nodes in the recursion tree. Ideally

31. (THEORY) Buyer-Seller and MAC electronic money transaction schematic:
-------------------------------------------------------------------------

        Buyer            A-------<id><refcnt:0>----------------------> Seller <id><refcnt:1> (increments refcnt)
        (<id><refcnt:1>  |
         <id><refcnt:0>  |
        after decrement  |
        refcnt           |
        )--------------->
Above has to be transactional (i.e atomic across cloud nodes)

32. (THEORY) MAC protocol reaper
--------------------------------
Reaper thread in each cloud node harvests the zero refcounted allocations and invokes destructors on them. Same  MAC id

33. (THEORY) Cloud Policing With Arbiters - Revisited:
------------------------------------------------------
When a suspect node is analyzed when a complaint problem is filed on it, (1) it is of foremost importance on how flawle

34. (THEORY) MAC Money Flow as MaxFlow problem:
-----------------------------------------------
Transactions happening in a cloud are edges between the nodes involved (buyer and seller). Thus it creates a huge direc

35. (THEORY) Cycles and components in above MAC Money Flow Graph:
-----------------------------------------------------------------
Above graph of money transactions could be cyclic which implies a supply chain. Strongly connected components of this g

36. (THEORY) STOCK TRADING:
---------------------------
One of the component in above MAC Money Flow Graph of cloud could be a virtual Stock Exchange. Based on the financial a

37. (THEORY) Analysis of Poverty and Alleviation through above money flow graph:
--------------------------------------------------------------------------------
Weights of the edges of money flow graph are the denominations of the transaction. Thus high value edges and low value

38.(THEORY) Demand and Supply and Value() function:
------------------------------------------
Alternative to the recursive definition of value() function above can be done through Demand and Supply - more the dema

39.(THEORY) Hidden or Colored Money:
-----------------------------------
In an ideal Cloud with only MAC currencies, colored money can co-exist if (not limited to) some money trails are missin

Total storage of money in Flow Market Graph = | Incoming money flow at Source - Incoming money flow at Sink | i.e Flow

There is a special vertex in Money Flow Market designated as Direct Taxation Hub which has incoming direct tax money fl

Colored Money = | Total storage money * Direct Taxation rate - Incoming Flow at Direct Taxation Hub Node|
Colored Money = | Incoming money flow at Source * Direct Taxation rate - Incoming money flow at Sink * Direct Taxation

Previous is an approximate naive zero-knowledge estimation of Colored money in Money Flow Market. This assumes that the

Above estimate is for direct taxation and resultant evasion. For indirect taxes (on Goods and Services etc.,), there is

Colored Money = | Total Goods and Services * Taxation rate - Incoming Flow at Indirect Taxation Hub Node|
Colored Money = | GDP^2 * Tax-to-GDP ratio - Incoming Flow at Indirect Taxation Hub Node| where GDP is assumed to be eq


References:
-----------
39.1 Network Flows over time over Storage Area Networks (SAN) - https://hal.inria.fr/inria-00071643/document
39.2 Network Flow - [GoldbergTardosTarjan] - http://www.cs.cornell.edu/~eva/network.flow.algorithms.pdf
39.3 Algorithmic Game Theory - Flow Markets - [TimRoughGarden] - http://theory.stanford.edu/~tim/books.html
39.4 RFID tagged currencies - $100 bill - http://www.businessinsider.in/New-Smart-Paper-Could-Put-An-End-To-Dark-Money/
39.5 Cons of RFID currencies - http://www.prisonplanet.com/022904rfidtagsexplode.html
39.6 Mechanism Design and Machine Learning - https://www.cs.cmu.edu/~mblum/search/AGTML35.pdf - Design of algorithms fo
39.7 Financial and Economic Networks - https://supernet.isenberg.umass.edu/bookser/innov-ch1.pdf

-----------------------------------------------
Commits as on 1 March 2014
-----------------------------------------------
Example java Publisher and Listeners that use ActiveMQ as the messaging middleware have been committed to repository fo
----------------------------------------------
Commits as on 17 March 2014
---------------------------------------------
KingCobra userspace library and kernelspace driver module have been implemented that are invoked 1) either in usermode
2) or through intermodule invocation through exported symbols in KingCobra kernel module, by the workqueue handler in V

---------------------------------------------
Commits as on 22 March 2014
---------------------------------------------
Minimalistic Kernelspace messaging server framework with kernel workqueue,handler and remote cloud client has been comp

----------------------------------------------
Commits as on 29 March 2014
----------------------------------------------
Initial commits for KingCobra Request Response done by adding 2 new functions parse_ip_address() and reply_to_publisher

--------------------------------------------
Commits as on 30 March 2014
--------------------------------------------
Both VIRGO cpupooling and mempooling drivers have been modified with use_as_kingcobra_service boolean flag for sending

--------------------------------------------
Commits as on 6 April 2014
--------------------------------------------
Fixes for REQUEST and REPLY headers for KingCobra has been made in virgo_cloudexec_mempool recvfrom() if clause and in

-------------------------------------------
Commits as on 7 April 2014
-------------------------------------------
New function parse_timestamp() has been added to retrieve the timestamp set by the VIRGO mempool driver before pushing

------------------------------------------
Commits as on 29 April 2014
------------------------------------------
Intial commits for disk persistence of KingCobra request-reply queue messages have been done with addition of new boole

------------------------------------------
Commits as on 26 August 2014
------------------------------------------
KingCobra driver has been ported to 3.15.5 kernel and bugs related to a kernel_recvmsg() crash, timestamp  parsing etc.

---------------------------------------------------
Version 14.9.9 release tagged on 9 September 2014
--------------------------------------------------
---------------------------------------------------
Version 15.1.8 release tagged on 8 January 2015
--------------------------------------------------


-------------------------------------------
Commits as on 17 August 2015
-------------------------------------------
KingCobra + VIRGO Queuing port of Linux Kernel 4.1.5 :
 - changed the REQUEST_REPLY.queue disk persisted queue path to /var/log/kingcobra/REQUEST_REPLY.queue
 - kernel built sources, object files
 - kern.log with logs for telnet request sent to VIRGO queue driver, queued in kernel work queue and handler invocation
KingCobra service request kernel function for the popped request; disk persisted /var/log/kingcobra/REQUEST_REPLY.queue

--------------------------------------------
Commits as on 14 October 2015
--------------------------------------------
AsFer Cloud Perfect Forwarding binaries are invoked through call_usermodehelper() in VIRGO queue. KingCobra commands ha

--------------------------------------------
Commits as on 15 October 2015
--------------------------------------------
- Updated KingCobra module binaries and build generated sources
- kingcobra_usermode_log.txt with "not found" error from output redirection (kingcobra_commands.c). This error is due t
- kingcobra_commands.c has been changed to invoke absolute path executable. With uncommenting of fd_install and set_ds

---------------------------------------------------------------------
Commits as on 10 January 2016
---------------------------------------------------------------------
NeuronRain KingCobra research version 2016.1.10 released.

----------------------------------------------------------------------------------------
NEURONRAIN VIRGO Commits for virgo_clone()/telnet -> VIRGO cpupooling -> VIRGO Queue -> KingCobra
- as on 12 February 2016
----------------------------------------------------------------------------------------
VIRGO commit:
https://github.com/shrinivaasanka/virgo-linux-github-code/commit/72d9cfc90855719542cdb62ce40b798cc7431b3d

Commit comments:
--------------------------------------------------------------------------------
Commits for Telnet/System Call Interface to VIRGO CPUPooling -> VIRGO Queue -> KingCobra
--------------------------------------------------------------------------------
*) This was commented earlier for the past few years due to a serious kernel panic in previous kernel versions - <= 3.1
*) In 4.1.5 a deadlock between VIRGO CPUPooling and VIRGO queue driver init was causing following error in "use_as_king
  - "gave up waiting for virgo_queue init, unknown symbol push_request()"
*) To address this a new boolean flag to selectively enable and disable VIRGO Queue kernel service mode "virgo_queue_re
*) With this flag VIRGO Queue is both a kernel service driver and a standalone exporter of function symbols - push_requ
*) Incoming request data from telnet/virgo_clone() system call into cpupooling kernel service reactor pattern (virgo cp
*) This resolves a long standing deadlock above between VIRGO cpupooling "use_as_kingcobra_service" clause and VIRGO qu
*) This makes virgo_clone() systemcall/telnet both synchronous and asynchronous - requests from telnet client/virgo_clo
either synchronous RPC functions executed on a remote cloud node in kernelspace (or) an asynchronous invocation through
 clause path to VIRGO Queue driver which enqueues the data in kernel workqueue and subsequently popped by KingCobra.
*) Above saves an additional code implementation for virgo_queue syscall paths - virgo_clone() handles, based on config
data passed to it either as a remote procedure call or as a data that is pushed to VIRGO Queue/KingCobra pub-sub kernel
*) Kernel Logs and REQUEST_REPLY.queue for above commits have been added to kingcobra c-src/testlogs/

-------------------------------------------------------------------------------------------------
Commits - KingCobra 64 bit and VIRGO Queue + KingCobra telnet requests - 17 April 2017
-------------------------------------------------------------------------------------------------
*) Rebuilt KingCobra 64bit kernel module
*) telnet requests to VIRGO64 Queueing module listener driver are serviced by KingCobra servicerequest
*) Request_Reply queue persisted for this VIRGO Queue + KingCobra routing has been committed to c-src/testlogs.
*) kern.log for this routing has been committed in VIRGO64 queueing directory
*) Similar to other drivers struct socket* reinterpret cast to int has been removed and has been made const in queuesvc

------------------------------------------------------------------------------------------------
(FEATURE-DONE) Commits - CVXPY implementation for Eisenberg-Gale Convex Program - 18 August 2017
------------------------------------------------------------------------------------------------
(*) First commits for Convex Optimized Market Equilibrium Prices
(*) Imports CVXPY Convex Program solver
(*) Objective function is a logistic variant of Eisenberg-Gale Convex Program i.e uses money * log(1+e^utility) instead
 money * log(utility) because of curvature error (log is error flagged as concave and logistic is convex per:
 http://www.cvxpy.org/en/latest/tutorial/functions/index.html#vector-matrix-functions)
(*) Formulates constraints and objective functions based on http://www.cs.cmu.edu/~sandholm/cs15-892F13/algorithmic-gam
(*) But, For all installed solvers ECOS, ECOS_BB, SCS, LS solved convex program prints value as None despite all constr
(*) Obviously it should have worked. Therefore this is only a partial implementation commit.
(*) This implementation uses numpy randomly initialized arrays for Money each buyer has and per-good utility(happiness)
(*) Replacing money with perceived merit values translates this Market Equilibrium - Intrinsic Value versus Market Pric
 Equilibrium - Intrinsic Merit versus Perceived Merit. This has been already described in NeuronRain AsFer Design Docum
        - http://sourceforge.net/p/asfer/code/HEAD/tree/asfer-docs/AstroInferDesign.txt and
        - https://github.com/shrinivaasanka/asfer-github-code/blob/master/asfer-docs/AstroInferDesign.txt

--------------------------------------------------------------------------------------------------
(FEATURE-DONE) Commits - Convex Optimization - DCCP - 21 August 2017
--------------------------------------------------------------------------------------------------
(*) import dccp has been added
(*) DCCP is the recent advancement and generalization of DCP for convex-concave programs
(*) method='dccp' has been added as parameter to solve()
(*) Objective function has been changed to log() from logistic() - curvature is concave which is in conflict with defin
eisenberg-gale convex program in textbooks. Reason for this contradiction is unknown.
```

```
(*) But DCCP overcomes the DCP limitation and solve() prints converged solutions for objective functions
(*) logs have been committed to testlogs/
(*) CVXOPT solver has been installed but it does not solve the Eisenberg-Gale objective function. Only SCS solver works
KKT conditions indirectly.

----------------------------------------------------------------------------------------------------------
(FEATURE-DONE) Commits - Convex Optimization - DCCP - 22 August 2017
----------------------------------------------------------------------------------------------------------
(*) Verbose set to True for printing Splitting Conic Solver progress information
(*) logs committed to testlogs/

----------------------------------------------------------------------------------------------------------------
(FEATURE-DONE) Commits - Convex Optimization update - 29 August 2017
----------------------------------------------------------------------------------------------------------------
(*) Removed hardcoded variable values in objective and constraints
(*) In the context of pricing, ECOS Error Metrics print the matrices of market clearing prices for goods
(Reference - pages 3072 and 3073 of https://web.stanford.edu/~boyd/papers/pdf/ecos_ecc.pdf - KKT conditions in ECOS sol

----------------------------------------------------------------------------------------------------------------
(FEATURE-DONE) Convex Optimization - Pricing Computation - 30 August 2017
----------------------------------------------------------------------------------------------------------------
(*) Prices of Goods/Services have been computed explicitly from Karush-Kuhn-Tucker Conditions (1,2,3 and especially 4)
(*) References:
        - Pages 106-108 of http://www.cs.cmu.edu/~sandholm/cs15-892F13/algorithmic-game-theory.pdf)
        - KKT conditions and Conic Optimization- https://arxiv.org/pdf/1312.3039.pdf
(*) logs committed to testlogs/
```