NeuronRain

Linux Kernel Forkoff with
CloudSystemCallPrimitives+MachineLearning+Analytics+Queuing+Debugging

Author: Srinivasan Kannan
(also known as: Ka.Shrinivaasan, Shrinivas Kannan)

http://sourceforge.net/users/ka_shrinivaasan
https://sites.google.com/site/kuja27/
https://github.com/shrinivaasanka

Subsystems of NeuronRain

- AsFer – BigData Analytics and Machine Learning
- VIRGO – New Linux Kernel with Primitives for Cloud (system calls and kernel modules)
- USBmd – USB Debugging module
- KingCobra – Kernel Byzantine Message Queue module

AsFer - Features

- Bayesian Classifier,
- Support Vector Machines,
- Decision Tree Classifier,
- String mining - Pairwise String Sequence Alignment (Needleman-Wunsch) ,
- String mining - Multiple String Sequence Alignment(BioPython,ClustalOmega),
- String mining - Powerset construction (C++) and String matching implementation in python(Jellyfish distance measures like Jaro-winkler, Edit-distance, Phonetic Matching rate etc.,) and
- Parsers and Encoding features for USGS and NOAA HURDAT2 data
- Automated Generation of Training and Test set for Classifiers that segregate the datasets into Event Classes and Automated generation of encoded astronomical object files specific to each Event Class.
- Sequential implementation of Discrete Hyperbolic Factorization and PRAM design for publication drafts in https://sites.google.com/site/kuja27/
- A Concept Hypergraph and HMM(Hidden Markov Model) based Experimental Inference Model Design inspired by Cognitive Psychology
- Design for extracting patterns from numerical data
- A Chaos Attractor implementation that generates a sequence of numbers using logistic equation and uses Python and R to compute a correlation coefficient between a non-Chaotic and Chaotic sequence of numbers.
- Discrete Fourier Transform computation for a parsed numerical input sequence using Python and R
- Spline interpolation, LOESS regression, Approximate Linear Polynomial interpolation and graph plot for a parsed numerical input sequence using Python and R (at present implemented for Dow Jones Industrial Average historical dataset and Riemann Zeta Function Zeros dataset)
- An Experimental Text Compression algorithm that uses Hidden Markov Model Maximum Likelihood Estimation based on a String Complexity measure that compresses vowels in text
- An Experimental Minimum Description Length implementation that uses the previous string complexity measure, computes MDL using Kraft inequality and Shannon entropy.
- Probabilistic Approximate Correct – PAC - Learning Implementation related to http://arxiv.org/abs/1106.4102
- Wagner-Fischer edit distance implementation for pairwise similarity of encoded strings
- An example implementation of Interview Algorithm and Intrinsic Merit Ranking for :
- http://arxiv.org/abs/1006.4458 and http://www.nist.gov/tac/publications/2010/participant.papers/CMI_IIT.proceedings.pdf
- An experimental Expirable Objects implementation
- Linear and Logistic Regression
- K-Means clustering implementation
- K-Nearest Neighbours supervised classifier implementation
- Linear Perceptron with Gradient Descent Learning
- Encoding of Maitreya Dreams to strings and some bugfixes for SWISS Ephemeris degree computation errors
- An experimental logical clock for cloud – EventNet – using Python pygraphs and C++ boost::graph with GraphViz rendering (from dot files)
- Longest Common Substring implementation for extracting common patterns in already classified and clustered encoded strings dataset
- Python script for translating the Pairwise Longest Common Substrings obtained from a cluster of Kmeans or KNN clustered astronomical dataset into rules that correlate location of astronomical entities to a class of events.
- Utility Knuth-Morris-Pratt String Match algorithm implementation
- Multipurpose bigdata analytics subsystem (storage abstraction on Cassandra/Hbase/Hive/File) for computing metrics from datasets before and after processing by above ML algorithms.
- Streaming Algorithms – LogLog and HyperLogLog Counter implementations for computing cardinality (distinct elements) in streamed multiset data
- Streaming Algorithms – CountMinSketch implementation for computing frequencies (heavy hitters) of elements in streamed multiset data
- Streaming Algorithms - Bloom Filter implementation for membership query in streamed multiset data stored in Apache Cassandra/Hbase/Hive/File NoSQL tables or disk and stream-simulated through python Generator/Iterable abstraction
- Pig script clients for Hbase/Hive and Python clients for Hive/Cassandra/Hbase
- Apache Spark Python RDD Transformation MapReduce script for mining Linux kernel logs and exporting the mined data as config to VIRGO kernel_analytics module. This integrates the AsFer or other Machine Learning algorithms into VIRGO Linux Kernel (with USBmd, KingCobra modules).
- Sequence Mining of ordered encoded strings (Apriori GSP Algorithm) for inferring Class Association Rules
- Named Entity Recognition using Conditional Random Fields and Hidden Markov Models (by finding Viterbi path through Viterbi Dynamic Programming Algorithm)
- Graph Search – Mine for Graph Relations (and render them in graphics) in Text Data by constructing the WordNet subgraph using the recursive gloss overlap algorithm in http://arxiv.org/abs/1006.4458 and http://www.nist.gov/tac/publications/2010/participant.papers/CMI_IIT.proceedings.pdf
- New Unsupervised text classification algorithm that computes nodes with prominent core numbers and PageRanks (nodes with high core-numbers and PageRanks are names of the classes) in the definition graphs obtained by Recursive Gloss Overlap algorithm above.
- Lambda Expression Compositionality from the depth first search closure of Recursive Gloss Overlap graph obtained from a text projected to WordNet.
- Social Network Analysis – Graph creation from social networking sites data (Twitter Followers etc.,) and their Psycho-Social-Sentiment Analysis (Bonacich Power Centrality, Eigen Vector Centrality, Recursive Gloss Overlap Graph Sentiment Belief Propagation and SentiWordNet based sentiment scoring etc.,)

USBmd – Features

- USB Debugging Kernel module

VIRGO - Features

- Config file support

- Psuedorandom Generator Based Loadbalancer

- Kernel space remote execution

- User space remote execution with kernel upcall and pthread creation of userspace library function or executable

- Example unit test cases

- Usermode output redirected logging feature for Kernel upcall to Userspace

- ntermodule Function Invocation in Kernel Space - through which any machine on cloud can be completely remote-controlled deep upto board and hardware cards through function names or commands sent through virgo_clone() calls.

- CPU Pooling Driver – Multi-kernel-threaded VIRGO cloudexec Kernel

- Driver Module for unrestricted service of virgo_clone or other client requests.

- Memory Pooling Driver (and a key-value store) - and system calls and userspace clients for it on Cloud nodes – virgo_malloc, virgo_get, virgo_set, virgo_free.

- Queueing Driver - that implements a wrapper over linux workqueue and also a native local queue - used for KingCobra requests queuing with handler invocation. Also implemented is a standalone kernel queueing service that listens on requests rather than being forwarded by CPU and Memory pooling drivers above.

- VIRGO Cloud File Systems Driver - that implements distributed cloud file system calls and telnet userspace clients for – virgo_open, virgo_close, virgo_read and virgo_write of a file on remote cloud node

- All the drivers above for CPU, Memory and File System on cloud have 3 paths each for telnet connection to remote driver kernel server socket and system call connection to remote driver kernel server socket – 1) parameter is executable in userspace 2) parameter is a function name which uses a kernel upcall plugin to execute in userspace 3) parameter is a function name executable in kernel space – configured by a boolean flag with in the driver binaries. Based on this flag either kernel upcall to userspace or kernel intermodule invocation is done.

- A config driver - for exporting config symbols

- Experimental Bakery algorithm kernel module implementation – for synchronization in cloud

- Utilities driver kernel module – a universal kernel module with exported utility function symbols that can be invoked across VIRGO Linux subsystems and Linux kernel including EvenNet logging kernel socket client.

- Experimental EventNet driver kernel service module – This listens on incoming EventNet log messages (Vertex and Edge) and writes to EventNet Vertex and Edge text files by VFS write. These files can then be massively processed by the boost::graph or pygraph GraphViz code in AsFer to create DOT files and graphics. Event vertices and edges can be logged by virgo_eventnet_log() utility function from any kernel module on any VIRGO cloud node.\

- From the above EventNet graph, a logical time ordering can be obtained on the cloud events and partakers which is for example useful in establishing money trail in KingCobra MAC eletronic money.

- Kernel Analytics Kernel Module – reads the config key-value pairs set by AsFer or any other machine learning software and exports them which can be looked-up in any other kernel module. The key-value config are analytics variables learnt by mining kernel or other logs and objects. With this an adaptive dynamic kernel which changes over a period of time depending on a machine learnt config is obtained. At present an Apache Spark usecase which mines kern.log and exports a config variable through kernel_analytics has been implemented.

Thus VIRGO has all the requisite minimum functionalities for a linux variant cloud, artificially intelligent, operating system

KingCobra - Features

A Byzantine Distributed Request Servicing Software with Queueing and Arbiters (that includes a new experimental Electronic Money Protocol – MAC (MessageAsCurrency) Design) on either Krishna iResearch intelligent Cloud platform or Hadoop Cluster

Implements a minimal kernelspace and userspace messaging framework using VIRGO cpupooling (virgo_clone) and memory pooling drivers that in turn queues the requests into a kernel workqueue. There is also a standalone VIRGO queue kernel service that listens on the requests without dependencies on VIRGO cpupooling and memorypooling drivers to forward the requests. The VIRGO workqueue handler pops the request from workqueue and invokes KingCobra driver's servicerequest exported function and replies to the publisher and optionally disk persists the incoming requests to filesystem through VFS. Differentiator is KingCobra implements a cloud messaging with a decentralized queue with disk persistence and workflow in kernel level so that hardware is easily integrated into cloud.

NeuronRain - Enterprise Analytics

- NeuronRain AsFer has implementations for almost all of the standard machine learning algorithms for bigdata analytics including streaming algorithms and implementation of algorithms in author's publications (in https://sites.google.com/site/kuja27/ and http://sourceforge.net/projects/acadpdrafts/files/)
- NeuronRain VIRGO is a new linux kernel fork-off that depends on AsFer machinelearning code and has new linux system calls and kernel modules.
- NeuronRain KingCobra is a queuing software that expands linux workqueues into a decentralized postoffice protocol message queue
- Above are in design and development since 2003
- Premium closedsource Enterprise Version of NeuronRain opensource is in development since 2010.

Unique Features in NeuronRain

- Probably the first opensource product that unifies linux kernel, cloud and machine-learning in lowlevel primitives instead of application layer – it is an IoT Linux OS Kernel.
- VIRGO Linux Kernel has new system calls for memory and file access across network through kernel-to-kernel socket communication only. No userspace sockets are involved (though upcall is supported to userspace through config).
- VIRGO memory RPC system calls implement kernel memory cacheing of data while filesystem RPC system calls realize persistence of data across network,
- VIRGO Linux Kernel is dynamic depending on key-value pairs learnt by AsFer machine-learning code from bigdata sets (e.g. logs)
- Supports invocation of new VIRGO Kernel system calls via Python application layer code in AsFer.
- New virtual currency implementation has been added in KingCobra based on Google Protocol Buffers – an alternative design to bitcoin – cloudwide object move - based on move semantics in C++
- Complete and updated list of features of NeuronRain is described in: https://github.com/shrinivaasanka/Krishna_iResearch_DoxygenDocs/blob/master/ProductOwnerProfile_With_FunctionalityDescription.pdf

NeuronRain – Applications

- VIRGO Can be used as an application deployment platform – by overlaying the kernel code base with VIRGO code and rebuilding
- Opensource Cloud operating systems can be deployed on VIRGO and the networking code in Cloud OS can be changed to invoke NeuronRain machine learning implementations, syscalls and drivers
- Existing linux system call code can be rewritten using VIRGO cloud system calls and is transparent to applications
- Presently OS kernels are static and with NeuronRain it becomes dynamic with machine learning.
- IoT and any device driver can be leveraged by the kernel analytics variables learnt by AsFer (C++ and Python)

NeuronRain repositories

- NeuronRain Research (academic, astronomy oriented) - http://sourceforge.net/u/userid-769929/
- NeuronRain Enterprise (generic, for production cloud deployments) -

  https://github.com/shrinivaasanka

Enterprise Usecase Examples

- Usecase1: High Network Latency and Bandwidth Allocation to high priority applications – AsFer analyzes webserver logs and learns analytics variables which are read by VIRGO kernel_analytics and exported kernelwide (or) writes key-values that can be modprobed and read from /sys.
- Usecase2: CPU scheduling priorities are dynamically revised based on AsFer-learnt key-value pairs in kernel_analytics.
- Usecase3: Loadbalancer in cloud which reads the AsFer learnt key-value pairs and dynamically updates the requests routing
- Usecase4: Self-healing OS kernel that heals itself by reading key-value pairs in kernel analytics done on oops dumps and panics
- Usecase5: Kernel Process Scheduler that realigns the queue based on kernel analytics key-value pairs
- Usecase6: KMEM allocator that realigns the heap based on kernel analytics key-value pairs
- Usecase7: Network Routers that reroute requests based on TCP traffic analytics
- Usecase8: IoT device drivers which dynamically program and recalibrate devices to the whims and fancies of kernel analytics key-value pairs
- Usecase9: CPU and Storage autoscaling based on kernel analytics done on the cloud logs.
- Usecase10: I/O scheduler that learns from kernel analytics key value pairs
- Usecase 11: IoT - Autopiloting automobiles and aeronautics
- Usecase 12: IoT – Intelligent Homes

NeuronRain Documentation and FAQ

- NeuronRain SourceForge and GitHub repositories have been documented in:

  http://neuronrain-documentation.readthedocs.io/en/latest/

- Detailed FAQ on NeuronRain and its licensing is periodically updated in previous ReadTheDocs documentation as features are added