
Matrix Os

/*==========================================================================
/
/ Matrix0s V0.1 Charge List By Jerome KASPER ©
/ All Rights Reserved. Copyright 2013/2014
/ Under Private Distributed License
/
==========================================================================*/

## *General Purposes :*

-› Matrix0s Project is a Teaching & Educational Project with full Documentary Components to complete Trainers for Developpers and IT Amateurs . Research / Development / Innovation / Proofs of Concept Purposes .

*No Discouragement Facing the Competition or Critics, Everyone makes his Way, Everything has Limits*

-› Elegant Design , Decentralized , with an Optimized Operating System Architecture in x86-64 AMD Assembly and a Lightweight Monolithic Kernel for Multi- Process / Multi- Processors / Multi-User Secure Use Model.

-› Near to Hardware and Quality of Service for Increased Performance and a Maximum User Experience through Innovative Design Concepts , a Restreind but Complete Hardware Support .

## *Kernel Core implementation:*

*Concept of "Self- Adjusted Spreaded GRID "*
-› The Consequent Increase in Volume and Speed of Main memories , lowered Price can let imagine a Model of " Continuous Shared Virtual Addressing " in a set of Computer Units and let forget the concept of local SWAP. HDD-Less Booting ( PXE ?) . Kernel Hot Replacement System ( replace / relink )

-› x86-64 AMD "Integrated" & " Low – Cost" APU Hardware Architecture Support for Proof of Concept Demonstration = › Multiplicity and scalability of the same Unit Type for optimum support, inexpensive and great potential.

-› Self - Adjusted Scheduling of the Whole System or in its Sub -Sets. Minimal Data Models and Effective Data Relations, Pointed as possible under Security Cover to avoid Useless Stacks of Protocols , unnecessary Data repetitions and / or Corruption
= › Effective Data Mass with Integrity Preserved , File Replication for Security purposes .

-› "Best Distributed Effort" Scheduling Algorithms : Virtualizing Resources Access , Asynchronous Load Charge Distribution, Parallel 'Pseudo Synchronous ' Process Threadind Spreading . ( Benchmarks Test on"Context switching" )

-› Smart System Approach in Time : Data Streams Enhancements / Organization , Scheduling of Data Access and Load Calculation time on the entire System .

## Distributed Self-Probed Scheduling "at Best ", Resource Reputation Propagation by "Peer Trusting", Organization Model :

-› Self Probing and Estimation by the System of it Self-Efficiency through " Time "/" Cycles" Temporal Measures System routines on Internal routine treatments , Regular Queries Procedures for creating the Resources Local "SpeedMap " ( State / Internal-External Inter- Resources Links Rates / Trust ) and " FillMap " ( Memory Types/Filling State/Read -Write Rates/Trust) and "System Stats" ( Statistics System of a Node / GRID and evolution = › log ) .
These Data Synthesis will be used for Scheduling Constraints in a GRID or a Node. Recursive And Iterative Evaluation Procedures (Total System evalution Calculation must not be ›5% Total System Load).

-› Decentralized Design with Smart Spreading of Calculation Mass and Data trough a Meta - Scheduler ( Tokened GRID Server or " Master Server " ) based upon constraints provided by Nodes Self-Probers ( Local / Remote work to achieve, somewhere else than on the Node if interesting ) . On a Local Gigabit Ethernet Internal Network for example, or an Internet connection such as ADSL/VDSL or Optic Fiber , it may be Useful to Schedule some "Instruction/+Data" Threads on "Network" to add a parallelism degree to Local Execution of a Process and therefore improve its Execution speed. There will be Computer Units or " Nodes", regrouped in GRIDs (VPN equivalent) multiple or not, giving access to a "Common Area Virtualized Memory Space" or " Shared GRID Memory " administered by a " Master Server ", " the most powerful " (Wich will lead to a "Cloud" Equivalent ) within a GRID . Everything will be the Accessible, in Addition to Local Resources .

-› The Resources "Processing Unit", "Memory", " File" , "Folder", "Tree", "Hardware Interface", "Software Interface", " Buffer ", "Processor ", "Settings ", " Bus "," Address ","Context", "Segment", "Directory","User" , "Node" and "GRID" are connectable (=› IDE ) and Organized according to a Relational Data Model . In each GRID , Data / Processings / Concepts / Hardware and Data Flows will be organized according to the Relational Model . During the Boot/System Installation, the System will proceed with the Initial Creation of the Node "Relational Tree " of "SystemMaps" . The Relational Model will facilitate later Evaluations of Local Node Resources and their Interdependencies / "Quality Weight" in each GRID .

## Resources "Meta-Weight" Attribution by "Peers Trust Propagation" with ((N + X)&&( X› = 2)) Maximum-Level Knowledge

( *No need to go too far , if everyone knows its neighbors is not bad :)* ) .

Each (N + X) "Neighnour Concept " Far-addressed in the " Shared GRID Memory Space " , Manipulated by the Whole System or in its sub-sets , is potentially subject to Failure / Corruption / Default / Slow etc ... .

The "Local" Trust Rate is used to evaluate the "Relative Quality" of a Virtualized Concept that you wish to access locally or in a Remote Way , according to a certain Point of View. The 'Global Mesh' establishment of the Local or General "Quality Weights" and Attribution of Resources becomes a "Small Repeated Effort" which will lead to a Global Better and Optimum Use of the System in Time .

The 3 Following Principles for "Peer Trust Propagation Establishing" in Local / Remote Resources will lead to the Global Calculation / Adjustment of the "Weighted Spreaded "Mesh"", as this block of instructions :

1 - Request a Resource Trust Weightvia the following 3 ways :

- Request directly Resource "Trust Weight".
- Request trough Confident proxy (Known random path)Local "Trust Weight" for Resource.
- Request via unknown proxy ( unknown random path)Local "Trust Weight" for Resource.

2 - Record Values Returned for each Resource(s) paths ( Value-adjusting for OutSourcing Time ) .

3 - Evaluation- Synthesis / Trust Weights Update / possible spoofs detection . Synthesis Return to "Master Server".(If necessary, a watchdog will kill "Dead " or " Insignificant" Resources (whose coefficient is around 0) + ( "Echos" and "Excessive Rebounds" Kills ))

-› Internal Scheduling Processing with "Execution Tickets" Attribution. Local or Remote Execution will be computed by (SpeedMap)+(FillMap) Scheduler Synthesis (if considered interesting or not) on a Node or on a "Virtualized GRID Resources". Scheduling will be done following a "Best Distributed Effort" Algorithm. Each CPU Core will be referenced as a "Processing" Resource with the associated stack " Will try to "Fill Out" the "Global Execution Stack" at most .

## Asynchronous Distributed Parallelism, Parallel " Pseudo-Synchronous" Execution and Secure "Virtualized / Transposed" Memory Accesses :

-› 'Distributed Asynchronous Parallel System' : in order to do not saturate any node in a majority of cases , we can parallelize following SpeedMap / FillMap the Execution of "Threads" of Code on Local Resources (Multiples Processors) and /or elsewhere in the GRID ( on other Nodes if necessary) while creating "Rally Synchronous Points" in the execution of the code, leaving temporarily Interruptions or other Instructions Running on a CPU core by example, Time to wait for "All Done and Back to Calling Base" or " Relocated " . These "Rally Points" will be " Grouped synchronous CallBacks " leading to the Execution of the Remaining of the Program when "Previously Launched Parallel Procedures" finishes . Need for Grid Global Timing Resynchronization of Local Context / Data / Instructions / after "Spreading" . Goal of "Pseudo-Real Time" Needed to make "Transparent" the Process Execution and its Fluidity ; Possible due to high frequencies Modern Buses But May Be a Side Effect to Control. .

= › This algorithm should avoid loads peak on Nodes by distributing the ' Excess Charge' on GRID(s) if "Overloaded" in Local. (Side Effect of "Overall Congestion ? " ) .

*Each node must accept a percentage of "External Load " to contribute to the "Global Effort" while favoring its Own Operations .*

## = › Parallel Distributed Multi-Core / Multi-Nodes on x86-64.

-› GRID Master Server is Decentralised trough a "Award -winning" Temporary System Token attributed to the "Best Node" according to Global Reputation (Best "note") of a node in the GRID( Voting Procedure (s) Master Server Nodes in a circle using the weighted Comparison System Stats and pre-allocation of "Additional Relief " N+Y Servers (Y › = 2) According to (N-Y) results of the vote . Such servers Reliefs will be in preventive "Starting Block"mode =› mini minimal daemon loaded on "Additional Relief " servers which, in case of fault of the main service, will let GRID "Minimum System Map" ensure a minimum GRID Load Scheduling to maintain continuity of service until the next Nodes Vote = › Fault Tolerance multiplied .

*"Pseudo- Synchronicity" Concept Developpement in GRIDs .*

In order to Manage "Asynchronous/Spreaded" Processings , the Master GRID Server(s) will regularly post Resync queries broadcasted to Nodes (Master Server TimeStamp Distribution to Nodes needs to be a regular Timing Based stuff, and if is there a Timing "Delta" on a node , Record the Delta Value or re-adjust System Clock). Assignation of a "Start TimeStamp" for each process, referring to the GRID Global Timer ( synchronized / updated ntp ?) .
"Pseudo - Synchronicity" is not an algorithm that Guarantee Results but recent Data buses are very Reliable and work at High Frequencies , the Results should be Convincing :) . Also , Timeout Check and Data IntegrityCheck will be intended to ensure the Quality of Data Transfers and the fluidity of the Execution Process ( Edge Potential Effect...) ("Relocation Process ? " ) .

-› Creating a Secure Remote Call System Procedure System for Asynchronous Parallel Processing on the GRID and the Load of "Threads" with " Data + Instructions " to Remote buffers for processing and return Result to Node making the Spread Request . Context of "Virtual External Jail" + Registries for Implementation of a Local "Remote" Segments .

-› Local Secure "Transposing Adress System" and Data Remote Access to Local "Node " Resources and " Shared GRID Memory" . Decryption Key available from the GRID Master Server in progress ( Supplied by Need , Ephemerally and Invisibly ) . Loading in "Local Secure Buffer" of Memory Adresses "Frequently Used " (Criterion ? ) To Avoid " Reroute" or abusive " Memory Adress Decoding " Regularly used ( + performance ) , while leaving it "Invisible" because this Table will be placed at a Randomly Memory Adress . If Data is Segmented , for Each use of a chained link, a "Reorganization" Ticket is assigned. Reorganization Process consists in putting (back) two segments (page (s) ) just near on a Memory Resource . Regular Reorganization will lead to effective contiguity on Supports Memory = › Enhanced General Memory Accesses.

-›Encrypted Memory Pointers Table and Local Resources for Kernel Self Verifications . Recurrent use of checksums (CRC) / Random-Systematics Audits on Critical Memory Segments to avoid Injections / Overflows / various Faults ... All Execution / Transfer " Sensitive " Data Processing will be systematically checked out 2 times at Start and Arrival in Buffers (Few 3GHz processor cyclesfor good security, is cheap paid ^ ^ ) A default callback procedure will apply by default in case of any error , delays or failed checks to restart Processing / Shipping / Verification of Bad data ( better to be safe =)) .

- › For the "Global" mode, there will be Z (Z › = 2) " Global Master Servers" whose purpose is to reference , Meta-Schedule and provide Competent and Efficient Subdomains Level 2 Servers, localized nearby physically, for an optimized response time . The " Network Trust Weights" synthesized by the " Global Master Servers" will guide Token (Attribution / Creation / Transfer) within local GRIDs( We still have time xD)

## File System :

- › Multiple Input Multiple Output System (RAID software with multiplied access ?) .

- › Support Mounting Currents FileSystems

- › Distributed File System Replication with Fault Tolerant Distributed ( RepMin › = 3) of Important and Used Files . Each file will be assigned a proportional "Redundancy Factor" .
= › More a file is Useful / Important , more the redundancy factor is important and so it will be 'more' replicated in the distributed file system (=› intrinsically more on each Node or more Times on a Memory Support at different locations ) . More recurrence = more Confidence not to lose a file, multiple local access , so faster, more "relevant " data and tolerance to losses

-› "Public FileSystem " dedicated to the Recurrency / Transfer of "Global" data on Local Areas Memory Nodes , or specific GRID whose Node belongs. Global Data Encrypted not readable "as is" on a Local Node without GRID Master Server Encryption Keys . Local split of " Global " data / "GRID Data" / "System Data" .

## User management :

-› Master Distributed and Shared "User Directory System" on GRIDs, managed by GRID Master Server and being publicly Encrypted in the same way as Transposed Addresses Table of Network Resource in a GRID, containing Permissions for the various User profiles of the different users on different GRID system Resources ( "White List System") . Copying Global User Permissions in Protected Memory Local Node in the Log .

-›Creating Profiles / Standards Users Permissions on a GRID ( User Profiles ) when Creating / Updating a GRID . User Profile Accesses and Other Rights Permissions (Standards) + Expansion / Truncation according Profile Rights assigned upon registration in a GRID .

## Mathematics improvements :

-› Pre-calculated Tables Loading System for Complex Formulas , Numbers Approximations in Kernel Space ( Trigonometry, Exponential , Infinity, 0 etc ... ) to induce more speed when computing basic mathematical routine on / Great and Small Numbers or Transcending Numbers according to the required accuracy .

-› NRNG hybrid based on non-sequential digits of PI calculated through BBP formula ( "Bellard " or " Viktor Adamchick / Stan Wagon" rewrite ) combined with internal entropy system and adapation of libc PRNG sources.

-› Native Management of (Compression / Encryption) for Data streams / files.

## Technical Standards to be Implemented :

-› Full 64-bit (long mode) Support . Other modes ( Compatibility / real / protected mode) Depreciated . 48 bits Total Adressing Space=256Tb.

-› Self-Probing : Highly Accurate Timer Daemon functions Debug (+ Registries Dumper ) Evaluation of Processing Cycles and TimeStamps with a a Standard "Read / Write" Query for example.
Resources Synthesis Algorithm by "Relative Weighting Establishment Map" .

-› Security : Local Complementary Encrypted and Verified Pointing System ( Pointers Table / Translation Address Tablein Kernel + Copy + Private Decryption Key distributed by GRID Master Server, placed randomly in memory for "at-Least" Dual-Verifications during Memory Access Requests ) . CRC Control Sums Management, Random/Systematic Audits on Memory Pages, Hybrid RNG , "Universal" Memory Addressing system 48bit-wide by Shared GRID Memory . Chained Threads Segmentation with Data Checkings based upon Chaining-Correspondence Table . Cascade Transposition of User Rights from a GRID "Master Server" to "Local" . "Invisible" Secure buffering of Recurring Used items ( addresses , tables , functions etc. ..) in Local Memory . AES + RSA (Accelered?) Encryption. Controls sums systematically inserted in the header file . Algorithm of "Forget" of Sensitive Data .

-› Network: Network Self-Adjusted Engine based on UDP queries ( performance + +) .
Ethernet implementation, 802.11 (TCP ?) , IPV4 , IPV6 , UDP , ICMP , ARP, NTP , DNS , DHCP, HTTP.

-› Hardware : x86 -64 AMD64 Native support , PCI, PCI- E, AHCI (SATA) , PS / 2 , USB, ACPI , VESA modes , NICs standard Intel / Realtek , Graphic AMD APU , "Standard" Driver Interface ? ( RegisterMap , DataFormats Methods ) .

-› Transcoding : Unicode 5.0 Support ( UTF-8/UTF-16 )

-› FileSystem Mounting System built to manage Standards on " External " supports ( Support of FAT32 local files , NTFS , ext2, ext3)

-› MatrixOs Specific Subdivisible FileSystem Optionally Distributed with "Distributed Network" and / or "Auto" and / or "System" Functions .

-› Connection System by Global Repository Translating (Address a "Resource" decoded through an ephemeral key transmitted by the GRID Master Server )

-› Inter-Process Communication By Context Transmission Settled to Security-checks (Context External Jail ) .

-› "Global" , " Undistributed " or " Private" Network Modes = › List of Nodes grouped according to a GRID-ID in "Private ", Local " Undistributed " and providing 2 addresses " Global Master Servers" in Global Mode according to the scope of desired connectivity.
Each node can potentially become Master Server in case of " Reorganization " of the GRID ( Following a Result-Change induced in a different Nodes Vote Result.) In " Undistributed " mode everything is managed at the Local level .

-› Processing algorithms optimized for AMD64 Assembler for Kernel / Base System Salls of the standard C library ( adapted / created ?) , Ultra -Fast Mathematics Arithmetic,Vectorial or matrix calculations ( SSE / SIMD Instructions etc. . ) .

-› Integration of a Standard C Library (Support Standard C11 ?) And (Adaptation / Creation? ) Of a Compiler associated to Superior Applicative Layer .

-› Differential Changes File Monitoring ( list diff instructions / Steps / Initial Context) to Manage Versions / intermediaries Snapshots on some data.

-› Standard Management Interface File Format ? ( Read / Write / Format Template / Matrix Input / Output Matrix )

-› Framebuffer Console .

-› System Software Update ( Via a system of " Snapshots Differential " )? .

- › Native Support of Accelerated 3D Graphics Library OpenSource ( WebGL , OpenGL , OpenCL ?) .

-› Native Integration of a SQL -Compliant Database System .

-› Adapting a Javascript Engine 2.0 (ECMAScript) . ( Userspace )

-› Adapting a HTTP System Browser / (X) ( HT ) ML / CSS ( WebKit ) ( Userspace )

## Supported file formats :

-› GUI : HTML , XHTML, XML , CSS, JSON , JS , TXT Formats . ( Parsing )

-› Compression: Standard sizes ( ZIP/RAR/TAR/GZ/Bz2 ) ( file compression)
+ "Kernel -sided " "Weighted Context" Algorithm (PAQ Data/Stream Compression-Decopression )

-› Graphics formats JPEG , BMP , PNG , TGA , SVG , PDF . (some Decode )

-› Audio formats WAV , OGG , MP3, MIDI . (some Decode )

-› Video formats MPEG4 , AVI , MPEG2 , DivX, x264 . ( Decode )

-› From DataBase Management : SQL Scripts

- › Assembly / C / Javascript

## Graphical User Interface / Integrated Development Environment / User Space:

-› GUI Application Management based on User Model and Web HTTP requests : no Window Manager but a " System Browser" Multi-Destops parsing CSS files / HTML / Javascript code / Images creating the appearance and Applications running under a Local Web format using the local Graphic Hardware Output ( " Server X" in the form of HTTP Local or Remote Web server) .
System Applicative Data on " relational SQL Database .
(Definitions of additional concepts Applicative Context , root dir or main () for javascript app ? , Architecture of a " Standard Application " ? Etc ...)

-› System Web Server with JavaScript ( Node.js syscalls ) accessible by a C Lib Managing the Treatment aspect of the code .

-› Implementing custom Node.js / V8 / Webkit

-› Application HTTPBrowser

-› Application GraphicalSystemIDE having three aspects :

1- Functional Graphs ( " Concepts " + links / O )
2- Interfaces Graphic Input / Output
3- Related functional Graph and Processing Listings .

-› Application Specific Web Connection User ( 'XDM ' Equivalent)

-› Application Desktop

-› Application Console

-› Application RelinkStreamPanel

-› Application FileManager

-› Application FileViewer

-› Application UserManagement

-› Application NetworkManagement

## Development Tools / Doc / Analysis

- › Creating an ASM Compiler X86 -64 AMD- compliant Web-Based Text editor as with Syntax Highlighting , Auto- Completion and Logically organized Library for the different explanatory instructions .

-› Creating a Graphic Simulation Tool of Processor Execution with Graphic Representation of the Execution of a Provided Listing and Registries / Battery / Memory and their respective values mode Footsteps.

-› Start Writing i18n Doc (Done!)