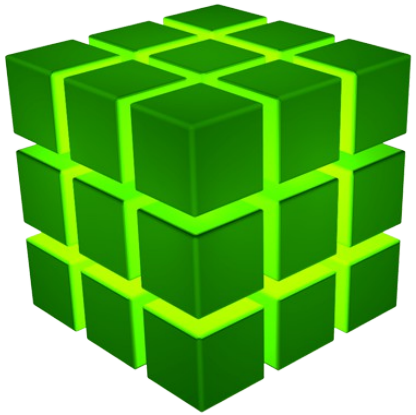


# Matrix Os

## Overview / Concepts

Matrix0s Model ©

# General Conception Lines : Global Abstract

- Assembly Concepted Fast Scalar x86 64 bits, 48bits Mappable-wide Memory Space, Multi-Node, Multi-Cores, Multi-Processus, Multi-User and Smart Repartited GRID Operating System.

- « Best Effort » Scheduling based on « Recursive Self-Probing » on Local and Global Resources, then (Re)Eval and Attribution . Asynchronous Distributed Parallel Scheduling with "Pseudo-Synchronous" Scheduled Execution.

- « Global » GRID Resources and « Memory Secure Mapping » . ResourceList with Concepts of Virtualised CPU, Memory and DataBus organised following « Quality Weight » and Standard Relations. Minimal Data Models.

- Security Oriented Conception focused on Cryptography, Multiple Data Integrity Checks, Multiple Other Verification Processes, Recursive Fault Tolerance with an Efficient Error Handling System and Secured « Remote Thread Execution ».

- « Virtualised/Transposed » Secure Flat Memory Access 48 bits-wide trough a Secure TranslationTable .

- GRID Distributed Mode : « VPN-alike » mode to join « Virtualized Resource Groups » Containing « Computer Units », refferred as « Nodes » to make Data Streams more Efficient through Peers.

- System Resource Evaluation by Peer Trust Process to Ehance Process Threads Attribution with « Quality Weight Evaluation», Indicator of « Global Resource Efficiency ».

Matrix0s Model ©

# General Conception Lines : Some Global Formulas

- Assembly Concepted Fast Scalar x86 64 bits, 48bits Mappable-wide Memory Space, Multi-Node, Multi-Cores, Multi-Processus, Multi-User and Smart Repartited GRID Operating System.

$$(X \in \mathbb{N} \wedge (X < 2^{48}))$$

$$\lim_{(X=1 \text{ to } 2^{48})} (GRID\ Master\ Server\ Adressing\ Space) = \sum_{(X=1 \text{ to } 2^{48})} (Memory\ Space) \in (Node(X) \in GRID) = 48\ Bits\ Linear\ Adressing\ Space$$

$$Node \in (1 \rightarrow X)(GRIDs)$$

$$GRID = \sum_{(X=1 \text{ to } 2^{48})} (((Node\ Resources) \in (Node(X)) \in GRID))$$

$$Process = \sum (Threads\ Blocks) \in Process$$

$$Scheduling = (\text{Fill Best}((\text{Resource}) \in GRID\ Weighted\ Resource\ List)\ with\ (Threads))$$

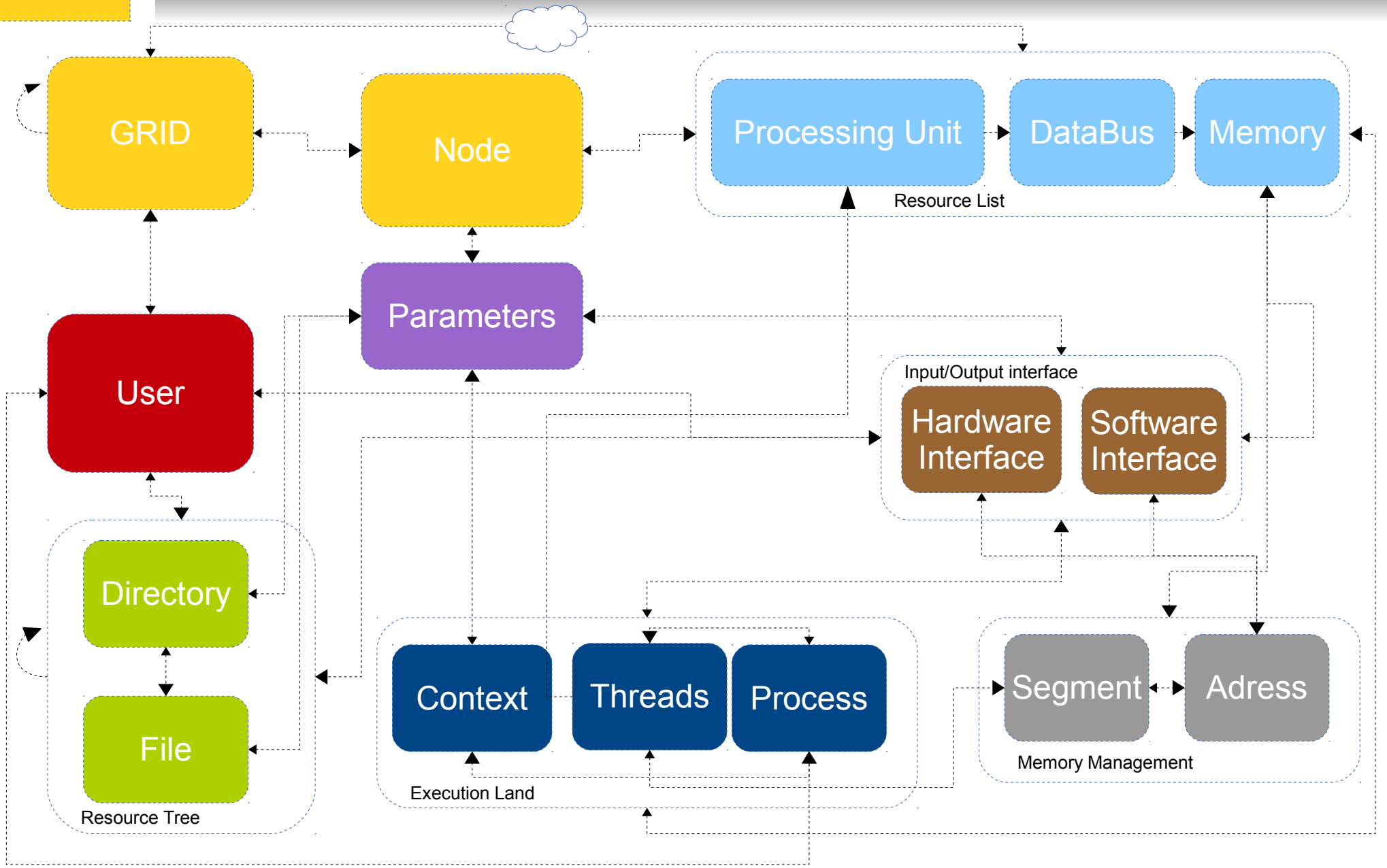
$$Process(ContinationStamp - StartSpreadingStamp) \rightarrow 0$$

$$GRID\ Charge \in (Instant_{(t)}) = \sum_{(X=1 \text{ to } 2^{48})} (((Threads\ excuted\ on\ Node(X)) \in GRID)) \in (Instant_{(t)})$$

$$Node\ Charge \rightarrow (\sum (Internal\ Node\ Threads) + (((Other\ Nodes\ Charge\ Excess\ Threads)/Total\ Nodes) * Node\ Weight))(Average\ model)$$

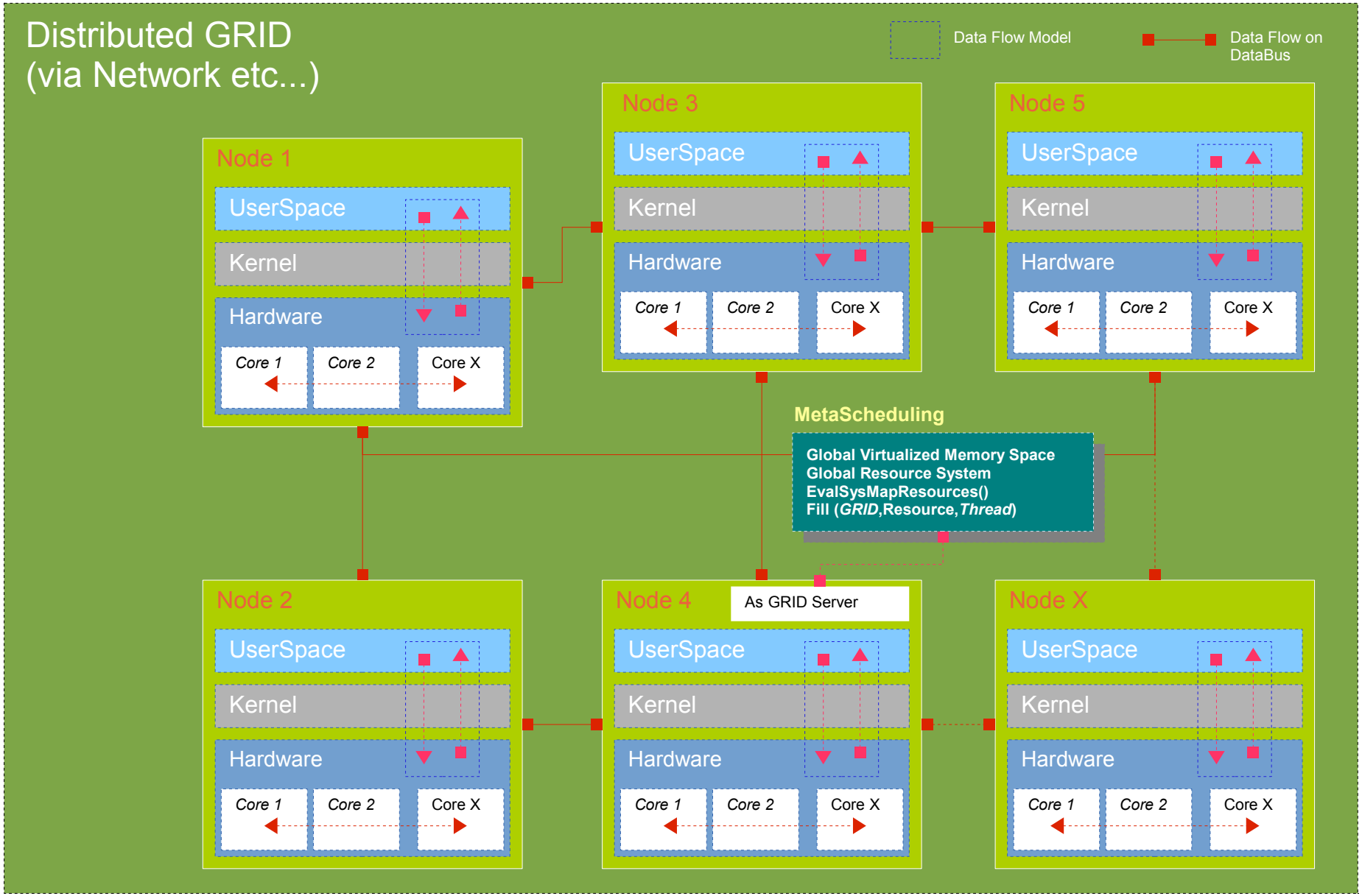
Matrix0s Model ©

# Relational Model between Concepts



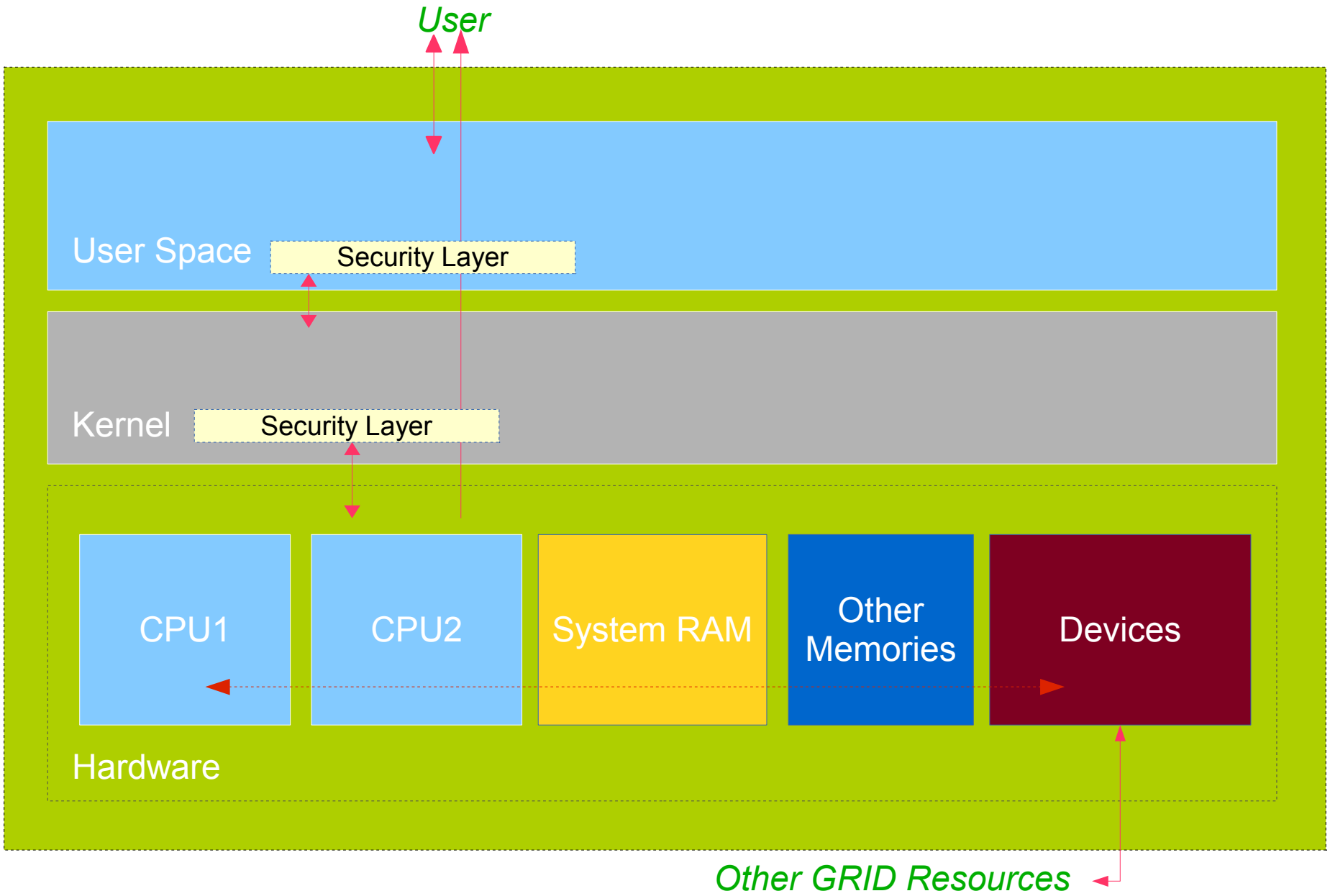
Matrix0s Model ©

# General Conception Lines : GRID Schematic



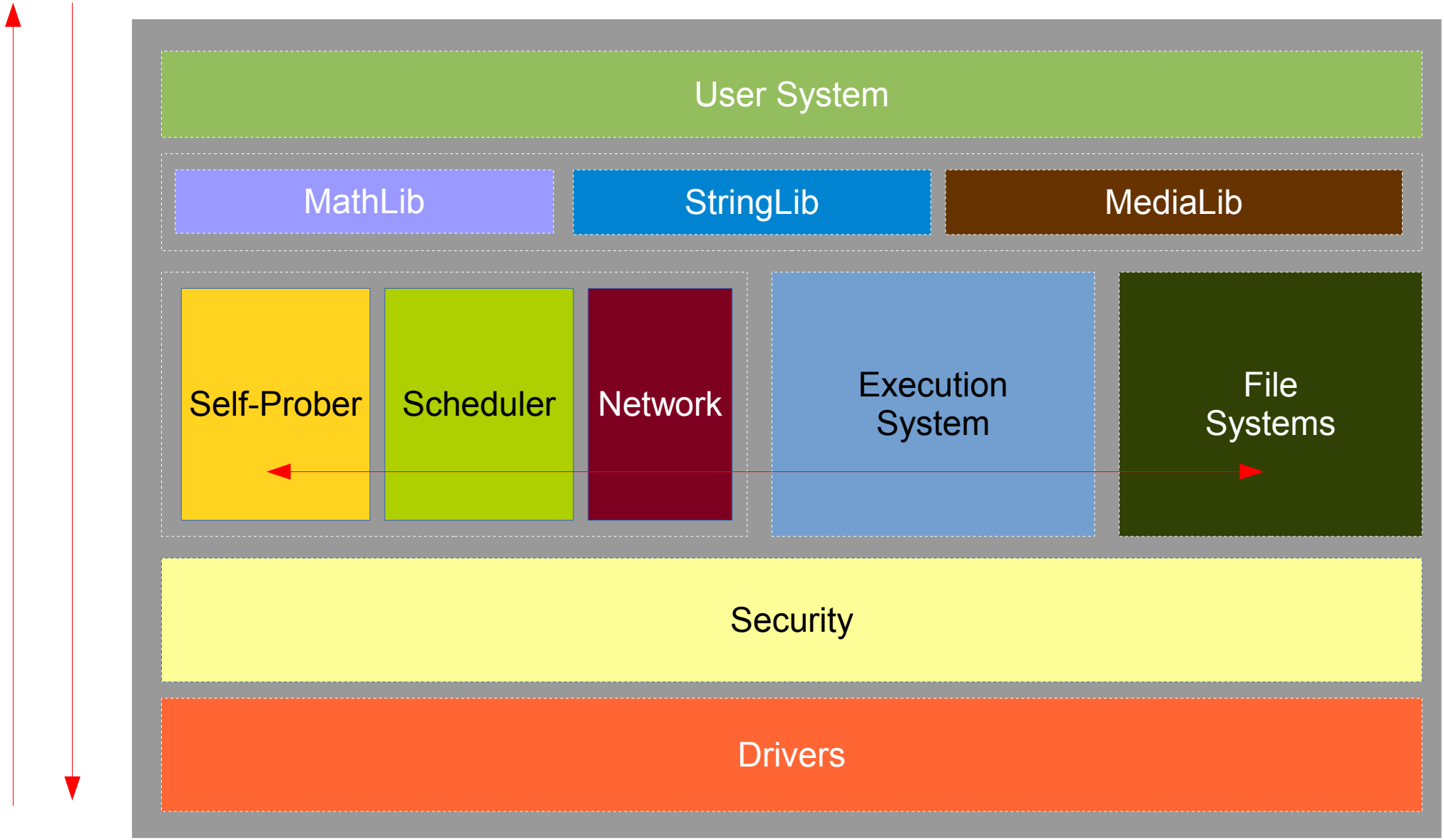
Matrix0s Model ©

# Detailed Node Model : Layered Model



Matrix0s Model ©

# Focus On Kernel Layers



Matrix0s Model ©

# Kernel Main Loop

```
BootPlace db 'KernelPath' ;
SystemRoot db 'SystemRoot' ;
; Provided By BootLoader

Load './Drivers/Drivers.asm';
Load './Security/Security.asm' ;
Load './ExecutionSystem/ExecutionSystem.asm';
Load './FileSystems/FileSystems.asm';
Load './Network/Network.asm';
Load './UserSystem/UserSystem.asm';
;Load All Generic Stuff

call 'KernelLoadMemorySecureAdressTranslation';
call 'KernelLoadMemorySecurePointing';
;Start MemoryAdressingModel

call 'KernelMapInitialLocalMemory', BootPlace;
call 'KernelLoadProtectedLongMode' ;
call 'KernelIntegrityCheck';
;Load Kernel binary and Check

if (IntegrityCheck=True)
{
  call 'KernelExecutionLevel',1;
  Load './SelfProber/SelfProber.asm';
  call 'KernelLoadResources' ;
  call 'KernelMakeSystemMap' ;
  call 'KernelDefaultConfigLoad' ;
  call 'KernelSystemConfigLoad', SystemRoot ;
  ; Enter Level 1 by establishing defaults

  if (SystemConfigLoaded=True)
  {
    call 'KernelExecutionLevel',2;
    Load './MathLib/MathLib.asm'
    Load './StringLib/StringLib.asm';
    Load './MediaLib/MediaLib.asm';
    call 'KernelLoadLib';
     ;Enter level 2 by by loading system "Lirairies"
    Load './Scheduler/Scheduler.asm'
    ;Memory Loacal Management
    call 'KernelLocalThreading';
    call 'KernelSetLocalUserMode';
    ;Finishing Starting up 'local' mode

    if(EnvCorrect=1)
    {
      call 'KernelExecutionLevel',3;
      call 'KernelLaunchDaemons';
      call 'KernelProbeNetwork';
      call 'KernelProbeRings';
       call 'KernelStartLocalSchedule';
      ;Enter level 2 by by starting Scheduling
      call 'KernelJoinRings';
      call 'KernelStartMetaSchedule';
      ;Join Ring if Possible and start MetaSchedule

      call 'KernelConsole';
      ;Start System Console
      call 'KernelGraphicMode';
      ;Start Graphic Mode
      call 'KernelProbeSystemRoot';
      ;Check For Main Start Point
      SystemRootApp db 'PathToSystemDesktop';

      while (State!=ERROR)
      {
        try{
          call 'KernelSCheduleExecuteStack',SystemRoot+'usr/apps/';
          ; Schedule execution Stack while ERROR
        }
        catch{
          call 'KernelRecoverFromError', ERRORMESSAGE ;
          ;if ERROR,try to recover
        }
        ; Kernel Main Loop
      }

    }
    else
    {
      call 'KernelReloadEnv';
      ;Reload Env until correct
    }

  else
  {
    call 'KernelDefaultConfig';
    ;Reload DefaultConfig until correct
  }

}
else
{
  call 'KernelReloadFromBootLoader';
  ;Reload Kernel until correct
}
```

Matrix0s Model ©

# General Conception Lines : Technical Implementations