Exposing and Exploiting Structure in Computation: A Unification Principle of Information Processing Systems

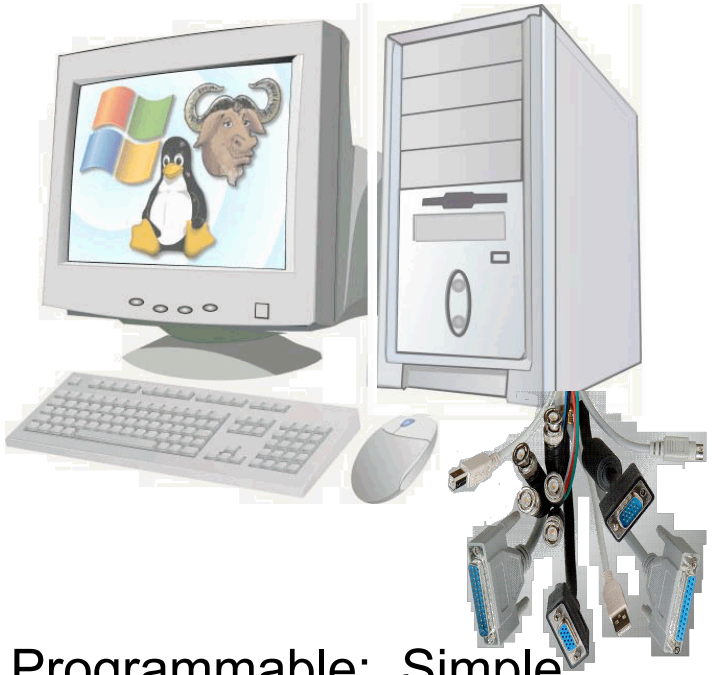
"Programmable Smart Machines"

DSRC/DARPA 2008 Summer Conference

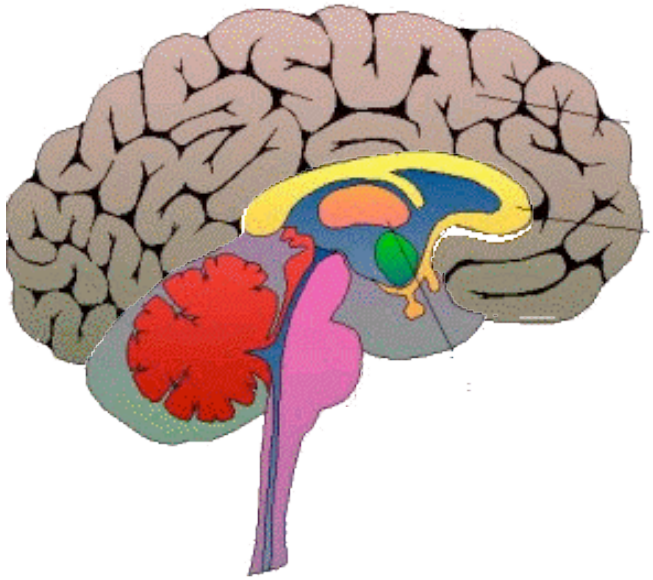
Jonathan Appavoo & Amos Waterland

IBM Research – T.J. Watson Research Center NY

Abstractly

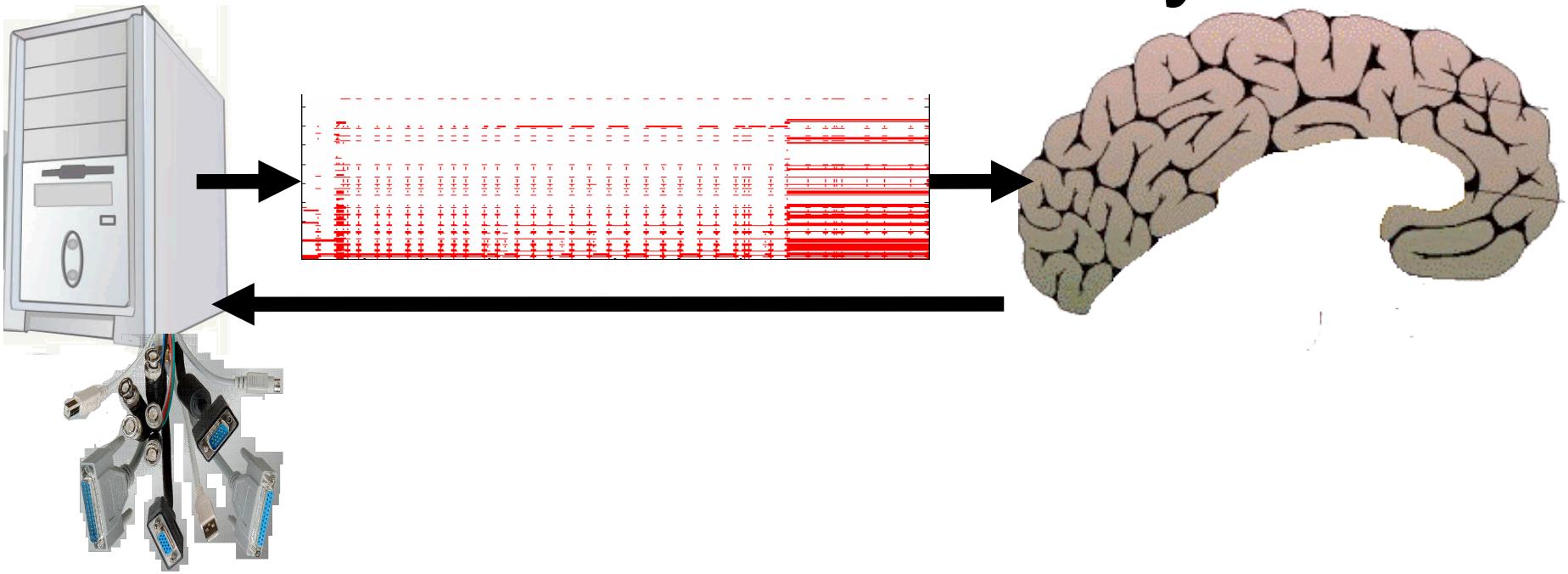


Programmable: Simple Layered Logical Expression.
Based on a simple Rational Machine Model.
Can be fabricated and interfaced using well know methods.
(Often overlooked but has an implicit model of time and more behavioural that most people realize)



Automatically learns and exploits structure in both space and time.
Maintains Global Context -- Constant Observation and Prediction
Flexible, Robust and Parallel by definition. Function scales with size.
Hierarchical:
- Multi-Scale structure
- Multi-modal Integration

The Crux → Novelty



1. Convert ENTIRE operation of a Modern Computer into an Execution Signal.
2. Extract, Represent and Express Structure in Signal.
3. Exploit Structure to potentially:
    1. Accelerate Future Computation,
    2. Expose Unknown/Un-Expected Correlations and
    3. Enable Extrapolation to generate new Behaviour.

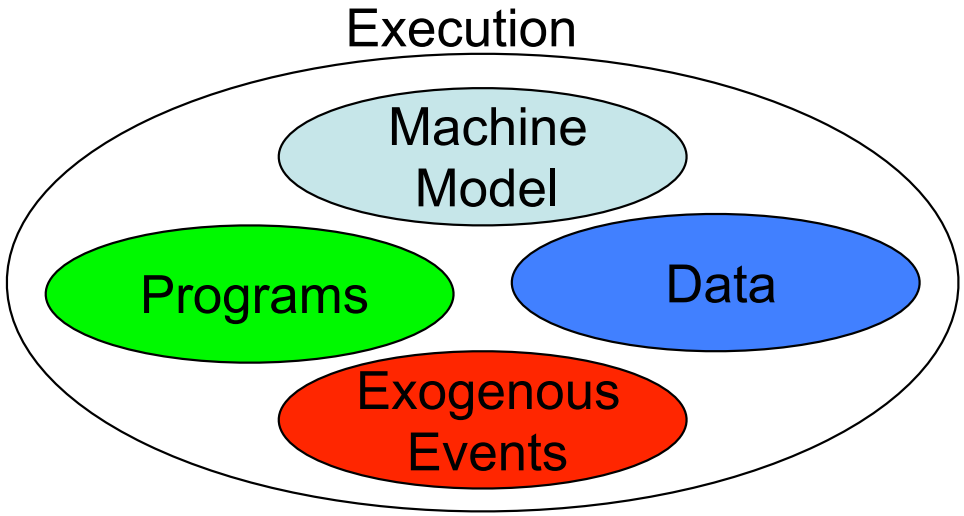
“Logic vs Statistics”

Mathematician David Mumford, “Pattern Theory: The Mathematics of Perception”

We do not believe it is “Logic vs Statistics”

Rather we conjecture that logical processes and statistical processes can be related to one another. And that this relationship speaks to a fundamental phenomena relating conscious logical mental processes with unconscious mental processes. This relation is a unification of information processing and suggests new hybrid computational models which exploit the relationship.

Meta Computer Science (MCS)



P(I) : Instruction Probability

P(D) : Data Probability

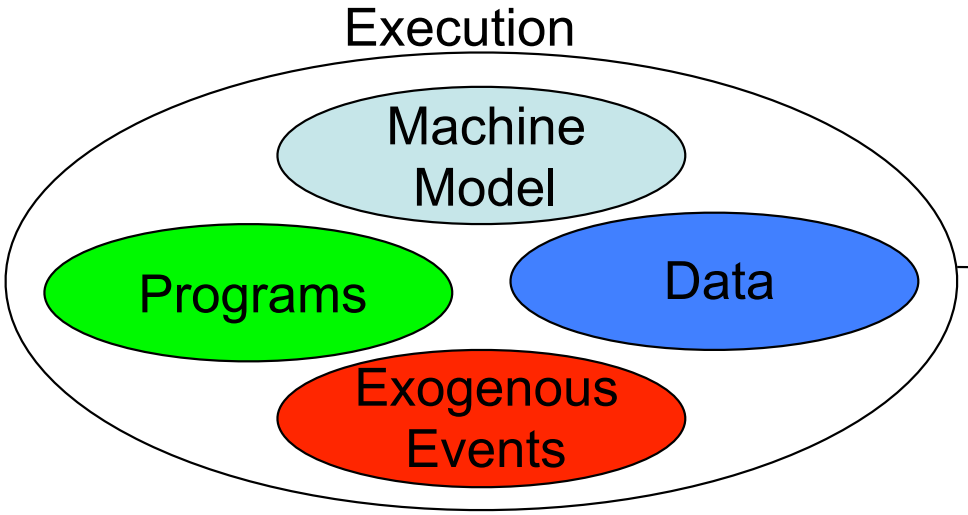
P(Ev) : Exogenous Event Probability

MM : Machine Model : Implemented in Hardware or by Software (simulator), has an explicit model for ordering (time)

Execution is the Joint of these probabilities given the Machine Model

Things that are reasoned about and concluded with respect to execution are independent of Specific Machines, Programs, Data and Exogenous Events.

Execution as Information



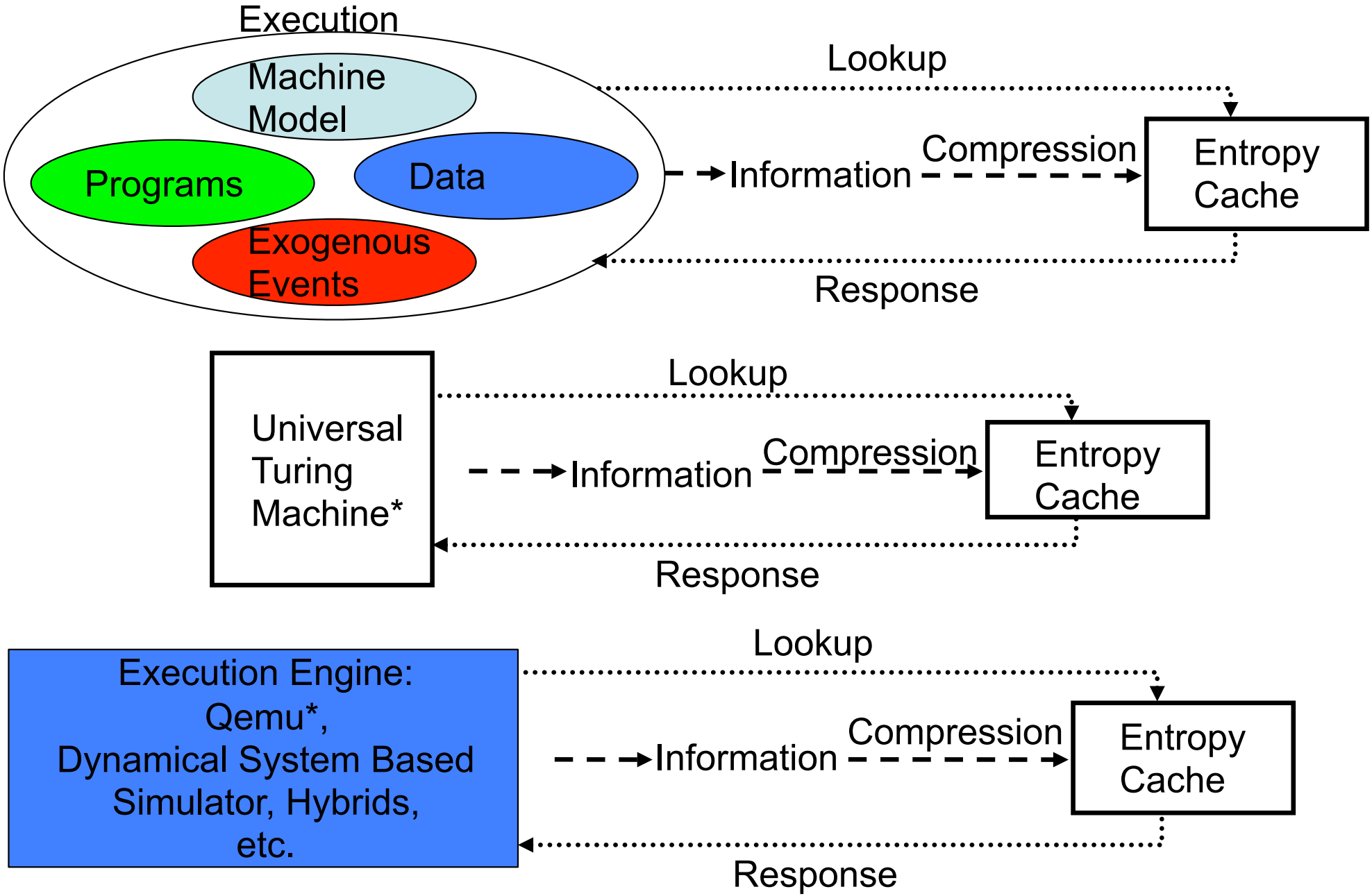
1101001001000100100001…

- Execution can be viewed as a process which describes a time varying signal.
- This signal varies based on underlying structural aspects of the sources of instructions, data, and exogenous events.
- This is a fundamental conversion in which each of the components of a traditional computer system is combined to yield the execution signal.
- Structure in the signal should be considered as a first class object for analysis and manipulation

It is our conjecture that:

1) Instructions are not random (there is structure in programs)
2) Data is not random (data ultimately comes from real world sources)
3) Exogenous events are not random (the sources in the real world have structure)
4) Significant structure exists that would allow execution to be compressed
5) Emergent structure can be exploited in ways that the system was not originally programmed for.

Meta – Machines : Inputers

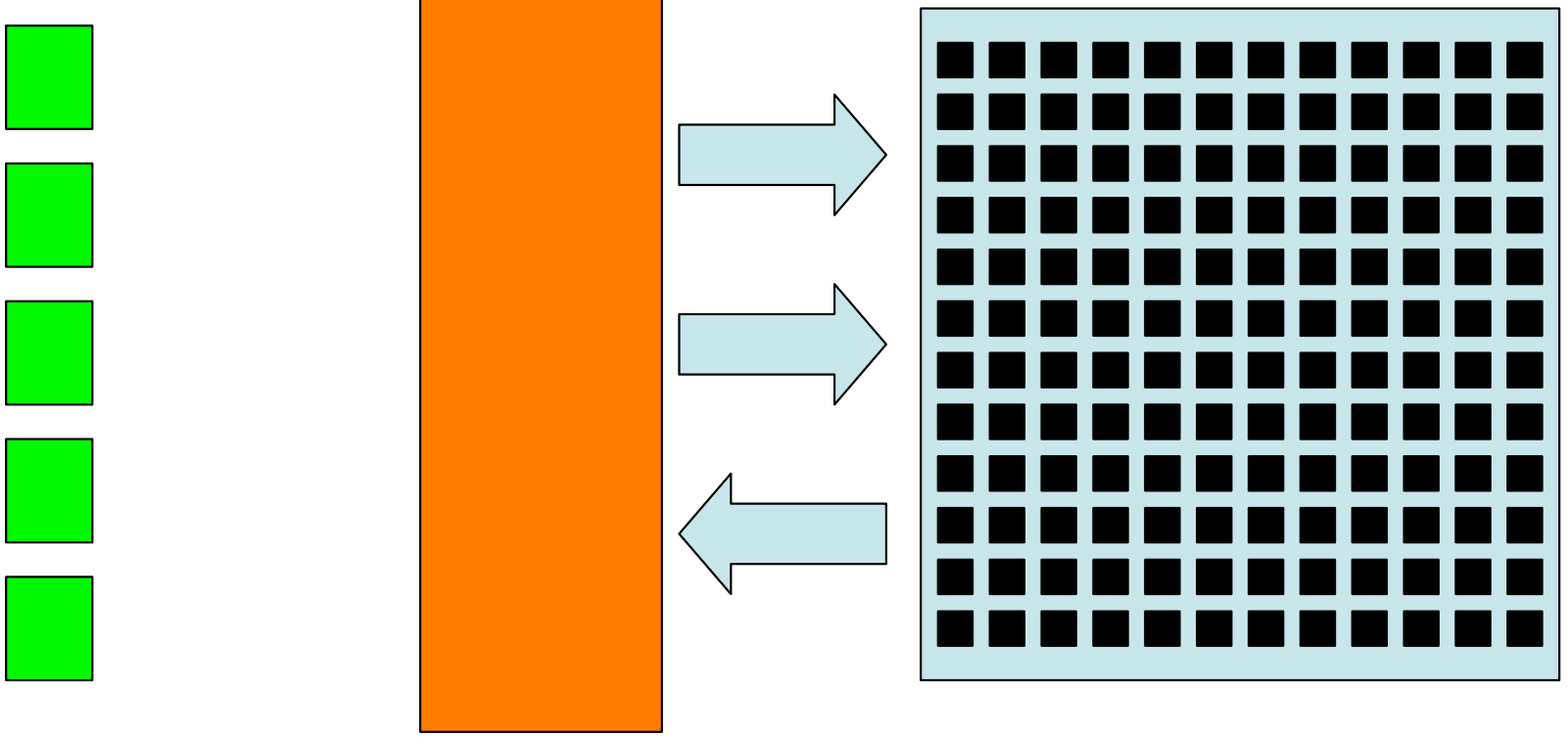


Abstract Model

General Purpose Computations Expressed as Complete Systems (All SW, External IO, and Machine Model).

Execution Engine

Entropy Cache:
Parallel Store and Search Fabric
Potentially Cache of states, Bayesian Graphical Model, etc.



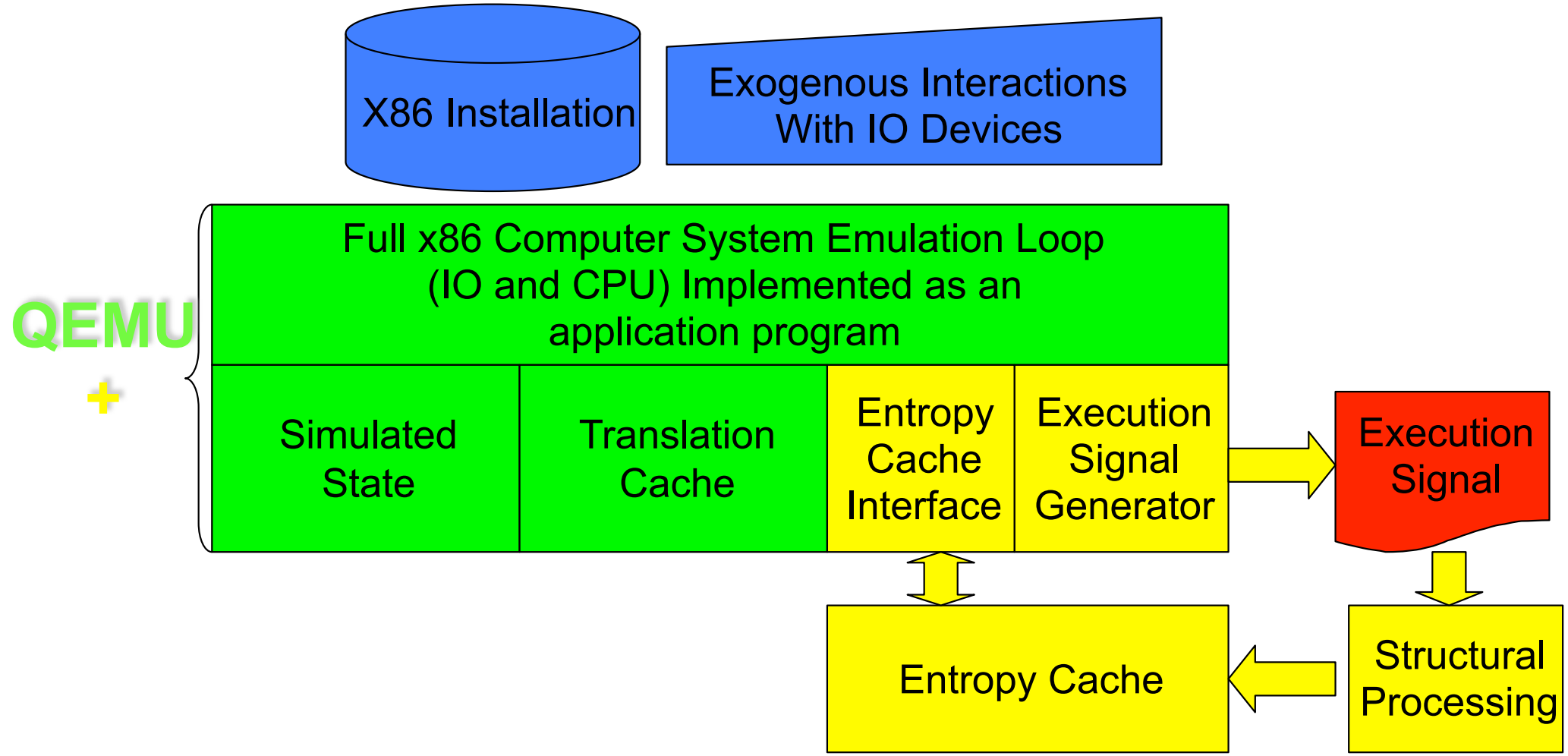
We are not looking to parallelize the execution but convert it into something that is parallel – recognition and retrieval of structure. Rather than thinking about how to program all the transistors we want to exploit the majority to learn, store and extrapolate on the structure emergent in the Execution. Leveraging the structure not only for acceleration but to enable extrapolation and identification of un-known correlations.

Execution Engine

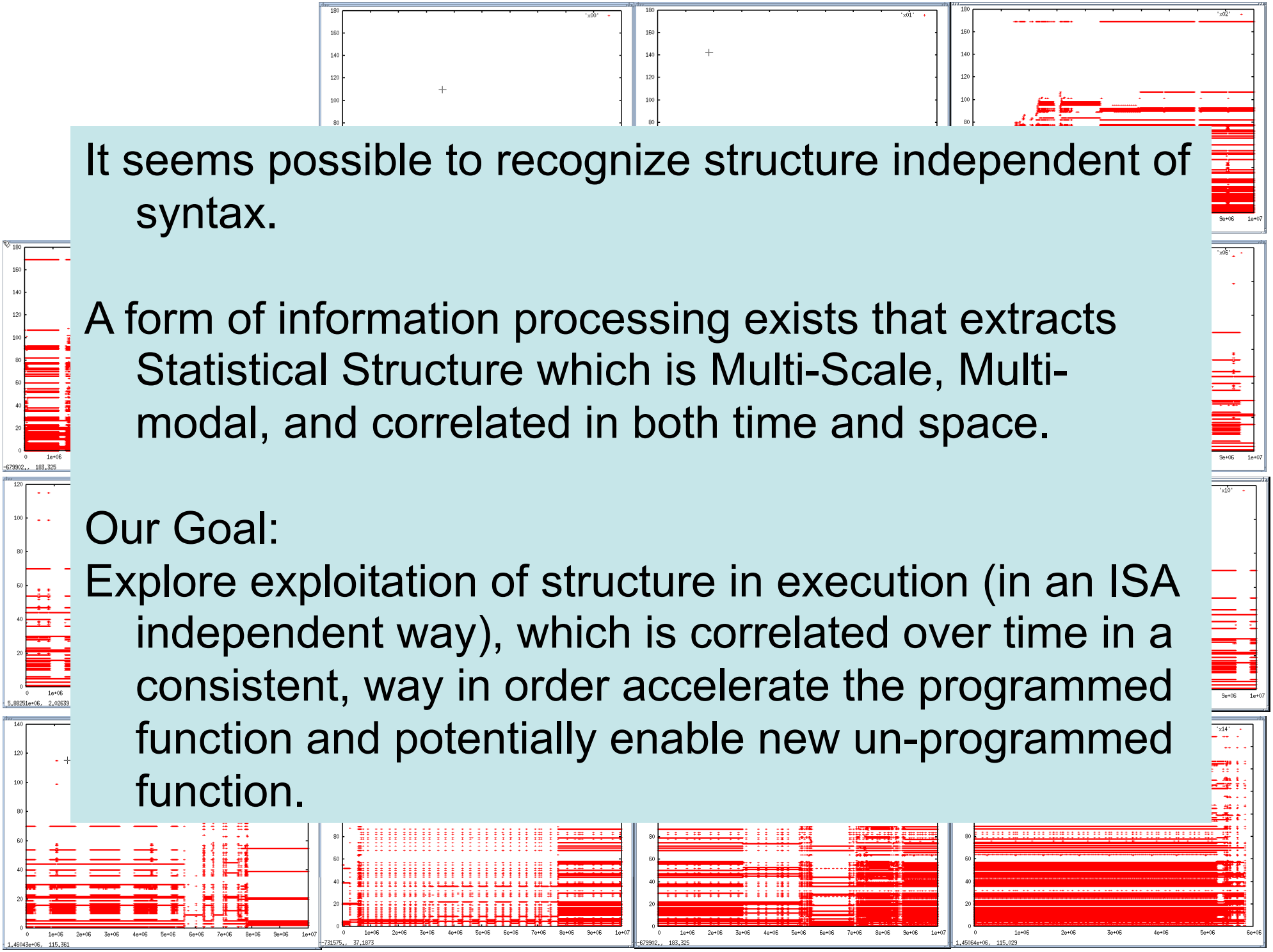
- Implement General Purpose (GP) Programmable Machine Model (eg. PC)
- Online conversion of execution into information for storage where information is not traditional instruction traces but rather encapsulates all structure in io events, instructions and data, generate from entire GP System being executed. Implicitly includes the Machine Model constraints.
- Basic Approach
  - Information is put into a global parallel shared store which can be queried and used to “accelerate” future execution (Execution caching).
  - Execution Engine queries parallel shared store to identify re-occurrence of structure and “advance” the execution.
- Advanced Approach
  - Integrates learning mechanisms including pattern extraction, recognition, prediction, and extrapolation (eg. speculation).
  - Interface to structural store which can be queried to utilize the correlations the system has learnt and utilize the structure to synthesize new execution.

Exploring the “Execution Signal” with a Classical Machine Model Emulator on BlueGene

Using an Emulator to Expose & Study Execution Signals of a Modern



+ a SuperComputer ;-)

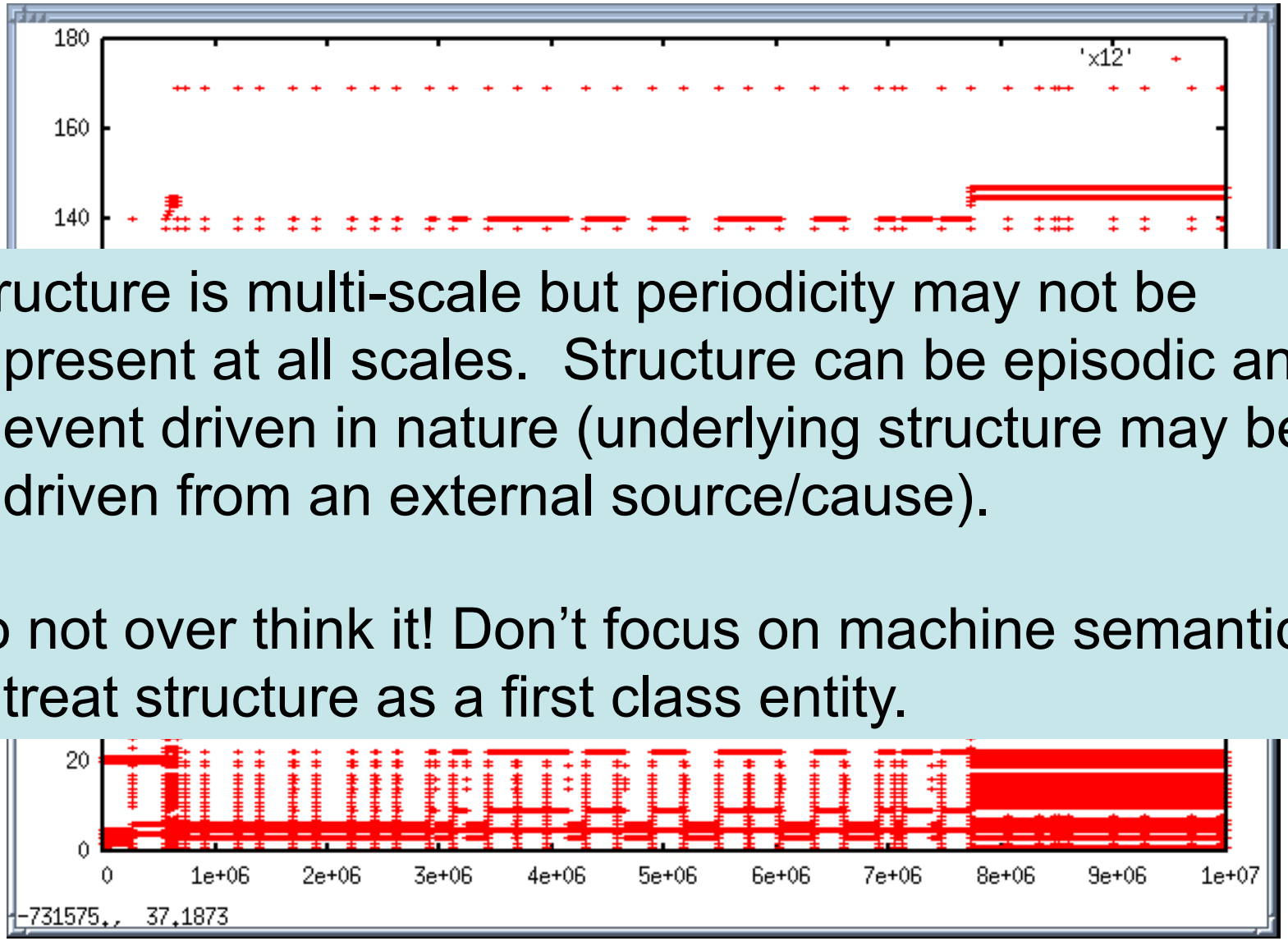


It seems possible to recognize structure independent of syntax.

A form of information processing exists that extracts Statistical Structure which is Multi-Scale, Multi-modal, and correlated in both time and space.

Our Goal:

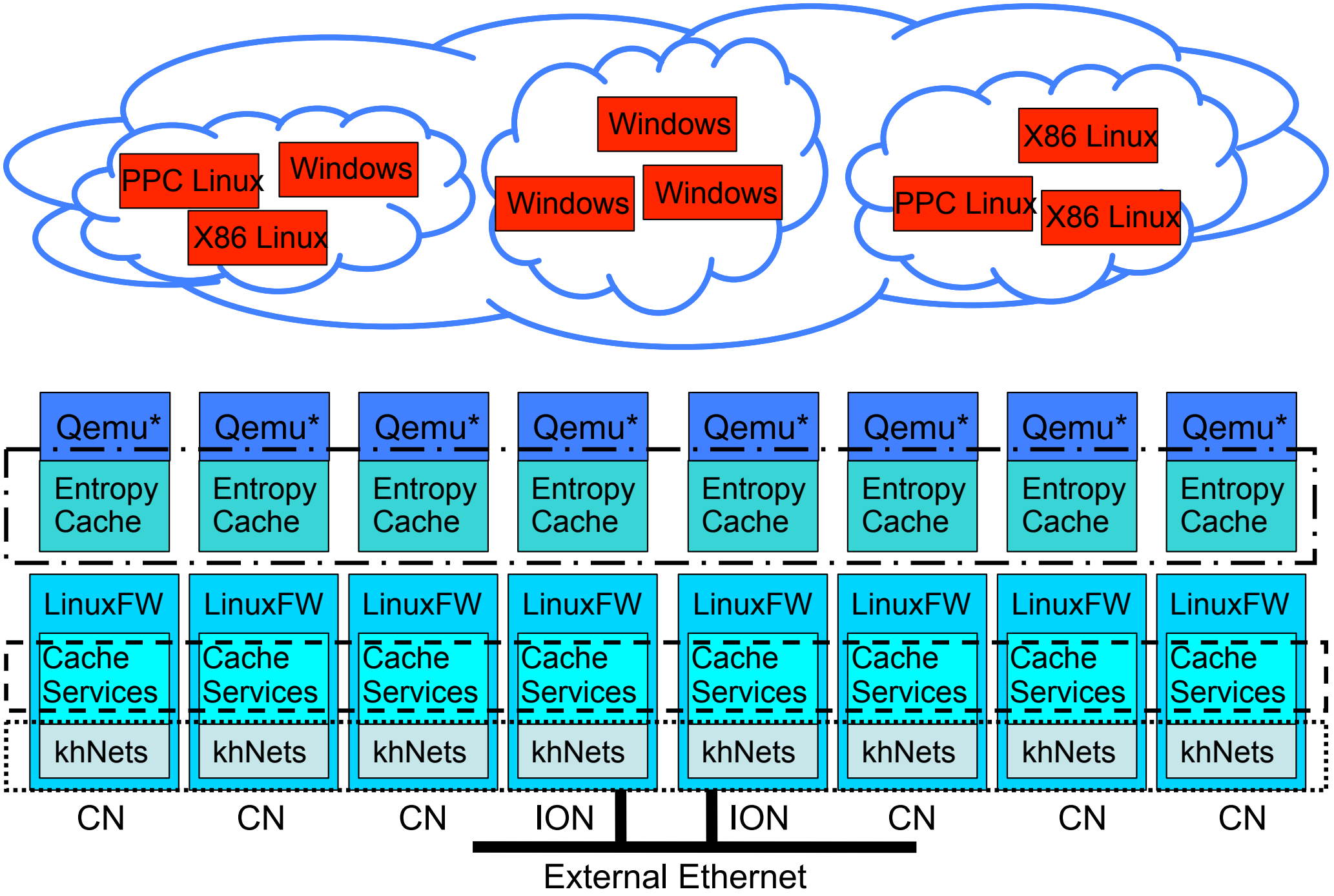
Explore exploitation of structure in execution (in an ISA independent way), which is correlated over time in a consistent, way in order accelerate the programmed function and potentially enable new un-programmed function.



Structure is multi-scale but periodicity may not be present at all scales. Structure can be episodic and event driven in nature (underlying structure may be driven from an external source/cause).

Do not over think it! Don't focus on machine semantics treat structure as a first class entity.

A Prototype Model Under Construction

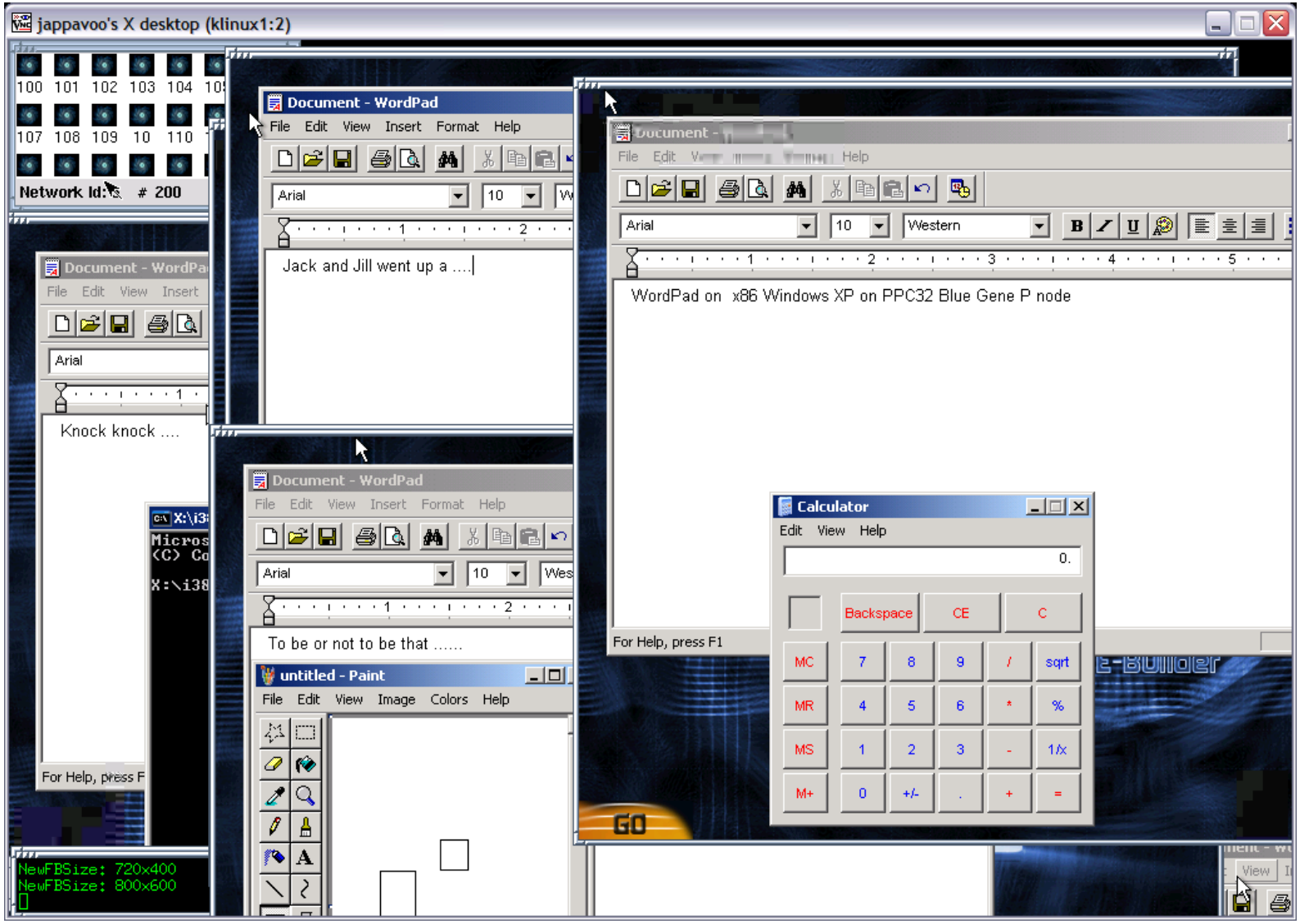


BG/P QEMU Demo

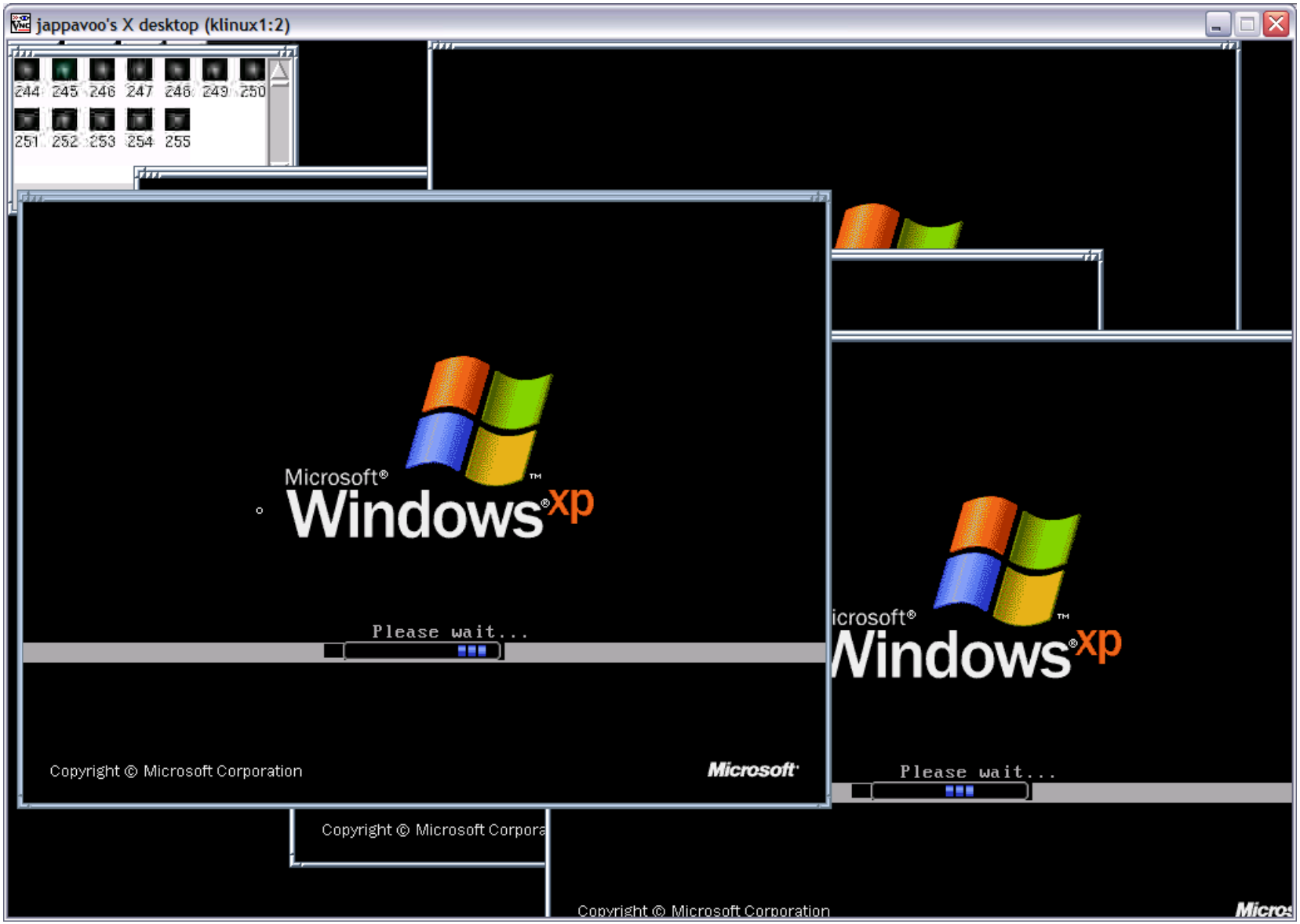
Goals: Given a full I/O Model and Timing Model

1. Construct Framework for Validation and Experimentation
2. Generate Data via controlled and uncontrolled experiments.
3. Prototype Hand Driven Structural Acceleration across instances.
4. Data for “Execution” Feature based sensor design for interfacing to Hierarchical Graphical Models such as HTM’s.

200 x86 Windows XP instances (800 PPC cores) on a common shared BG/P “QEMU-” Infrastructure (plenty of cores left 3296 + ****)



Booting Windows XP on BG/P “QEMU-” : FULL AND DIRTY Environment



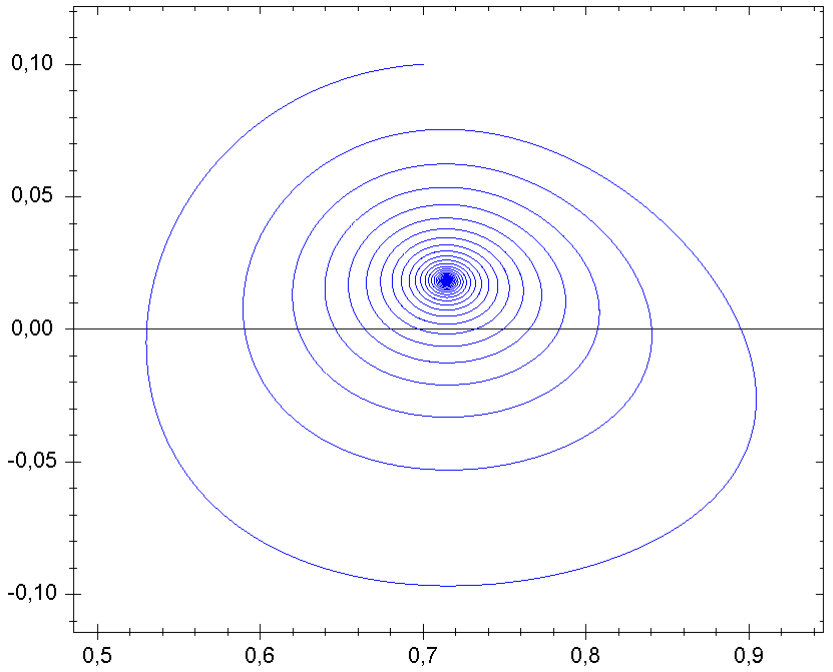
Dynamical System Based Simulator Designed to Explore Exploitation of Structure in

A Dynamical System Execution Engine

Developed a mathematical formalism (based on the physics notion of a dynamical system) for expressing execution which allows us to construct an x86 ISA compatible machine which utilizes lookup as its core primitive for resolving/advancing -- LEAPING -- execution forward in the phase space. Natural and simple way to exploit the deterministic structure in Execution via relationship between locations in phase space and Machine Model implementation.

Stored-program computers are dynamical systems

- A dynamical system is a phase space combined with an evolution rule
- A phase space is an object in which every state of a system can be represented as a unique coordinate
- An evolution rule is a function that maps a point in the phase space to another point in the space
- Integration is self-composition
- Evolution rule is integrated
- Offers an enduring denotational semantics



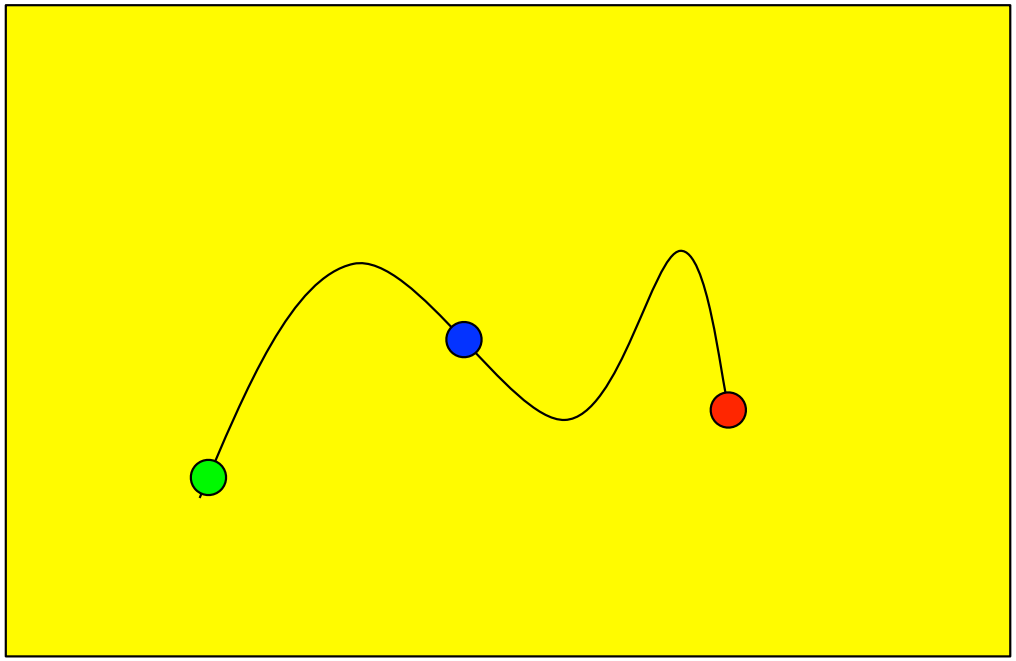
Deterministic Structure Manifests in Phase Space → Predictive "LEAPING"

Every point in phase space is a fully specified Machine Model state. Past Execution Drescribes Orbits given a set of determinism constraints.

If the Machine Model's current state corresponds to a point on a known Orbit then we can simply LEAP the machine state to the future point determined by the Orbit.

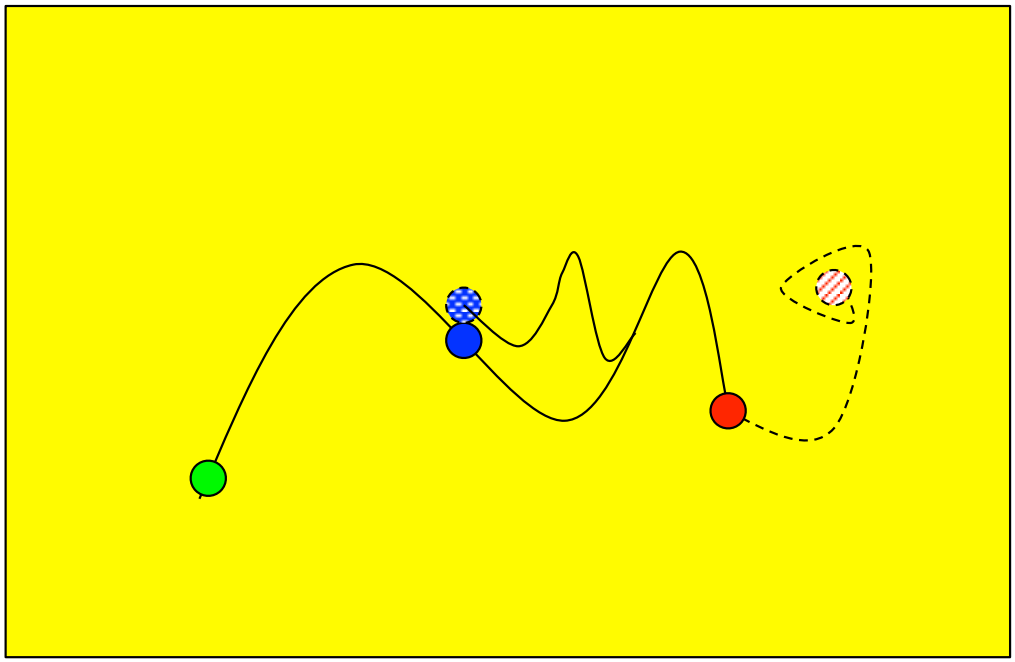
Orbits and the likelihood of reoccurrence is a description of structure in the Execution Signal.

Entropy Cache → Cache of Known Orbits



Look up / Search is Parallel Operation

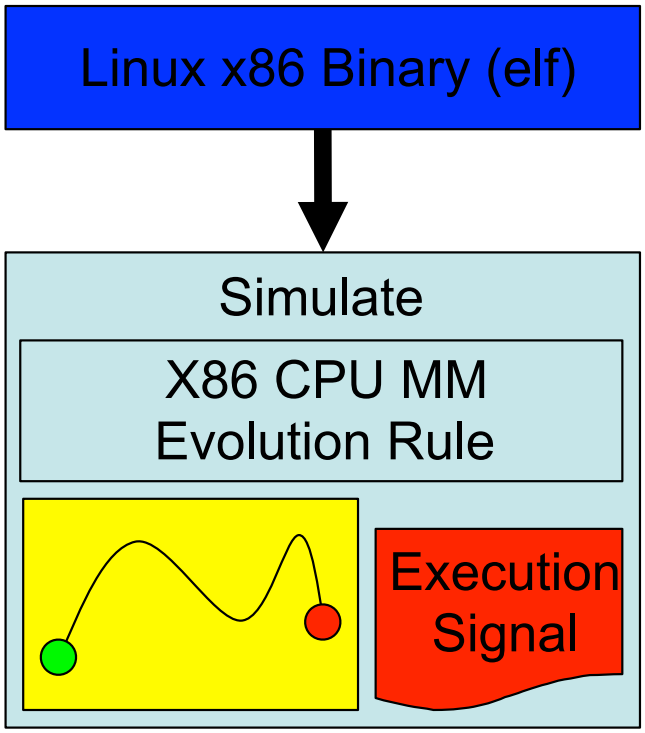
Extrapolating Execution in Phase Space → Speculation as Orbit Exploration

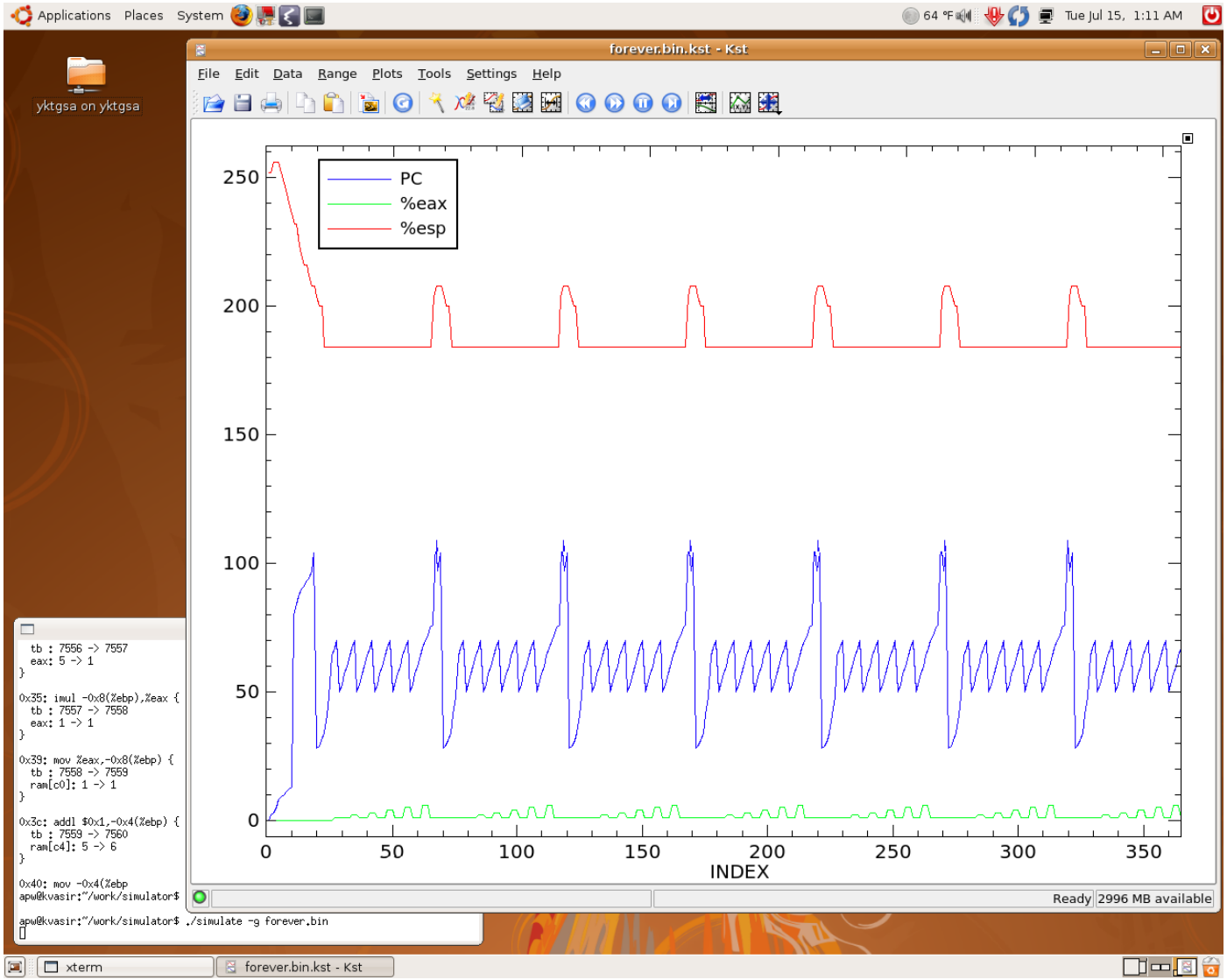


Speculation is trivially Parallel. Given enduring denotational semantics of orbits all future execution can benefit from orbits determined and cached.

DEMO/Status

“Simulate”

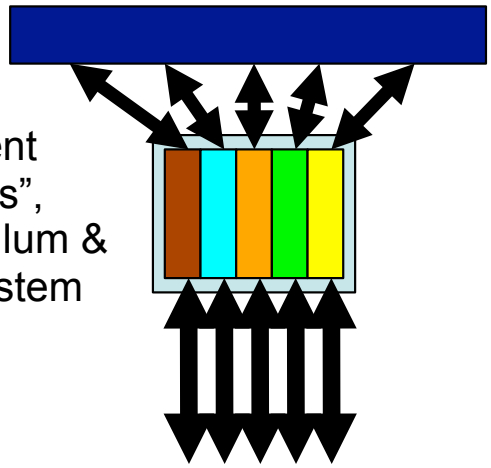

Linux x86 Binary (elf)
Simulate
X86 CPU MM
Evolution Rule
Execution
Signal


Applications Places System
Tue Jul 15, 1:11 AM
yktgsa on yktgsa
forever.bin.kst - Kst
File Edit Data Range Plots Tools Settings Help
PC
%eax
%esp
INDEX
Ready
2996 MB available
xterm
forever.bin.kst - Kst

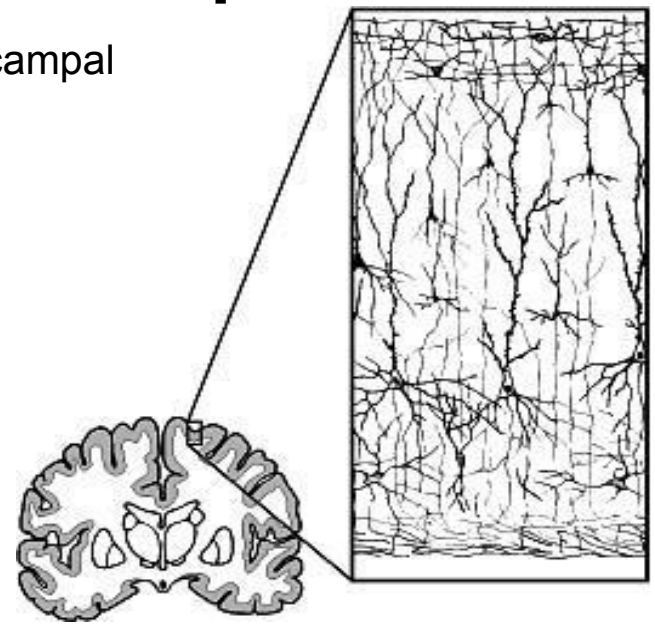
Cortical Computational Models and Advances -- Inspiration, Influence and possible

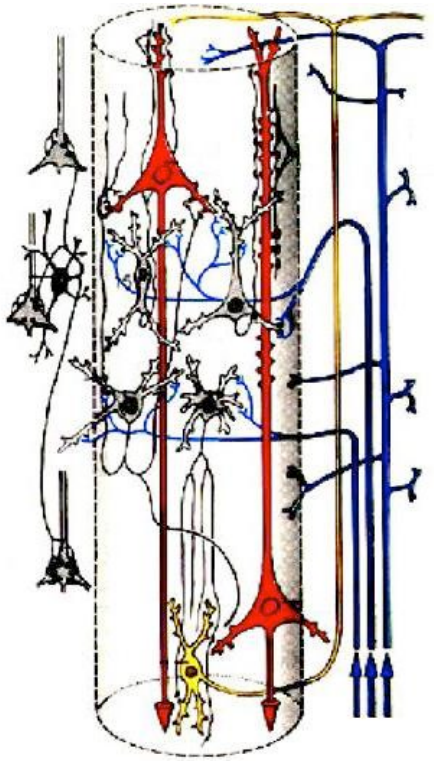
Why is there hope that the statistical structure can be recognized and exploited?

Telencephalic circuits (90%)
cortex, striatal complex, hippocampal formation, amygdala nuclei

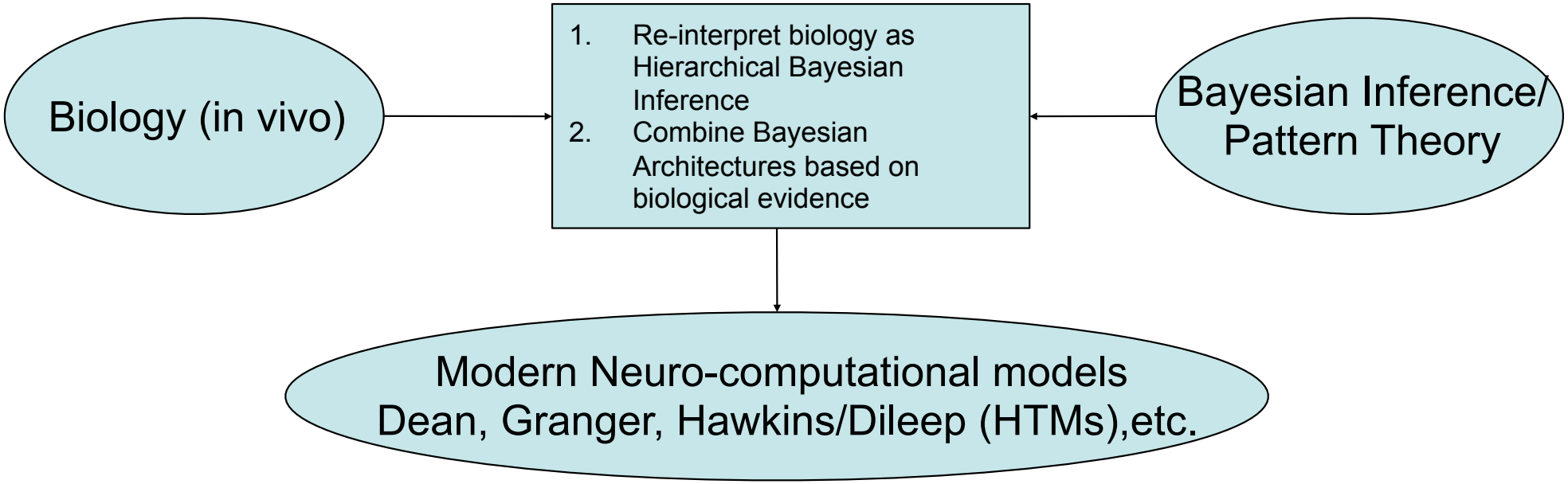


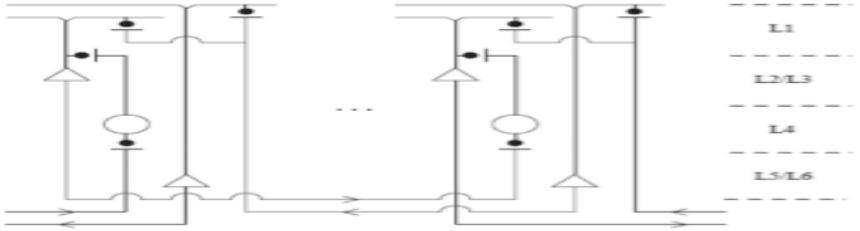
"ancient circuits", crebellum & brain stem





"The cortex functions as a very robust associative memory and computing device. It integrates data from different senses, different times and different resolutions, allowing us to identify correlations across time and across sensory modalities. Not only does the cortex enable us to robustly recognize patterns in the midst of noisy backgrounds, but we can do so even if we have never encountered a given pattern at a particular scale, orientation or brightness or in the midst partially occluding distractions. The cortex continuously predicts what we are likely to perceive next and warns us if events run counter to expectations."
Thomas Dean, Brown University 'Computational Models of the Neocortex'.





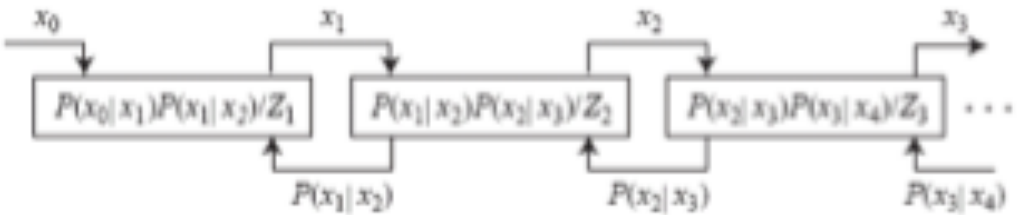
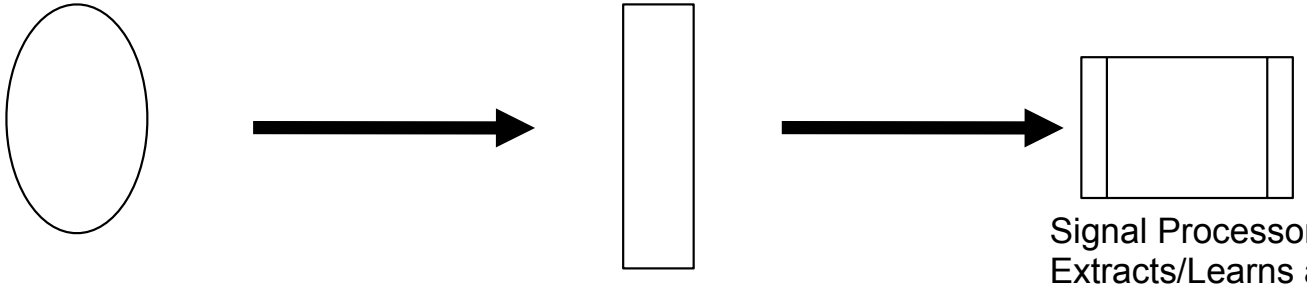


Table 1: Summary of derived brain circuit algorithms and initial citations

Circuits studied	Functions derived
thalamocortical core loops	clustering, hierarchical clustering (Rodriguez et al., 2004)
thalamocortical matrix loops	sequences, chaining, hash codes (Rodriguez et al., 2004)
striatal complex / basal ganglia	reinforcement learning, exploratory action selection (Granger 2005)
cortico-striatal loops	automatization; variation; power law (Granger 2005)
amygdala nuclei	filters / "toggles" (Parker et al., in prep)
cortico-amygdala loop	state-dependent storage & retrieval; category broadening (")
hippocampal fields	time dilation / compression (Granger et al. 1996)
cortico-hippocampal loops	spatiotemporal relations; navigation (Kilborn 1996)

Researchers propose that the recognition and storage of commonly occurring sequences and sequences of sequences are a central role of neural systems and this ability is can be modeled with Bayesian hierarchies.

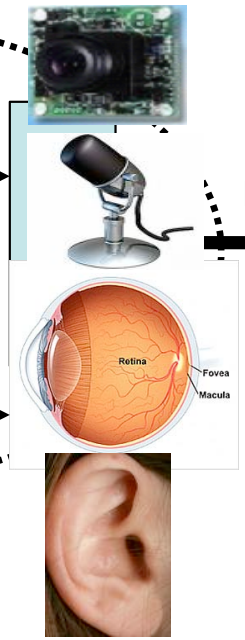
A Machine Learning/Biological Interpretation of Execution



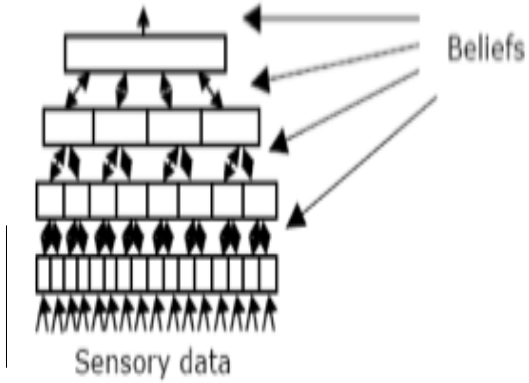
Cause/Source:
With some stable deterministic properties which are invariant under some “useful” conditions.

Sensor/tranducer
(signal generator)
Observers source via some carrier which is constantly changing and generates a compatible signal for down stream processor

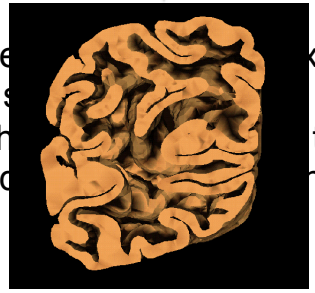
Signal Processor:
Extracts/Learns and Stores relationships in signal which are deterministic, invariant and correlated both in space and time and potentially at multiple scales. Corresponds to invariant features/objects in source.



Execution Signal

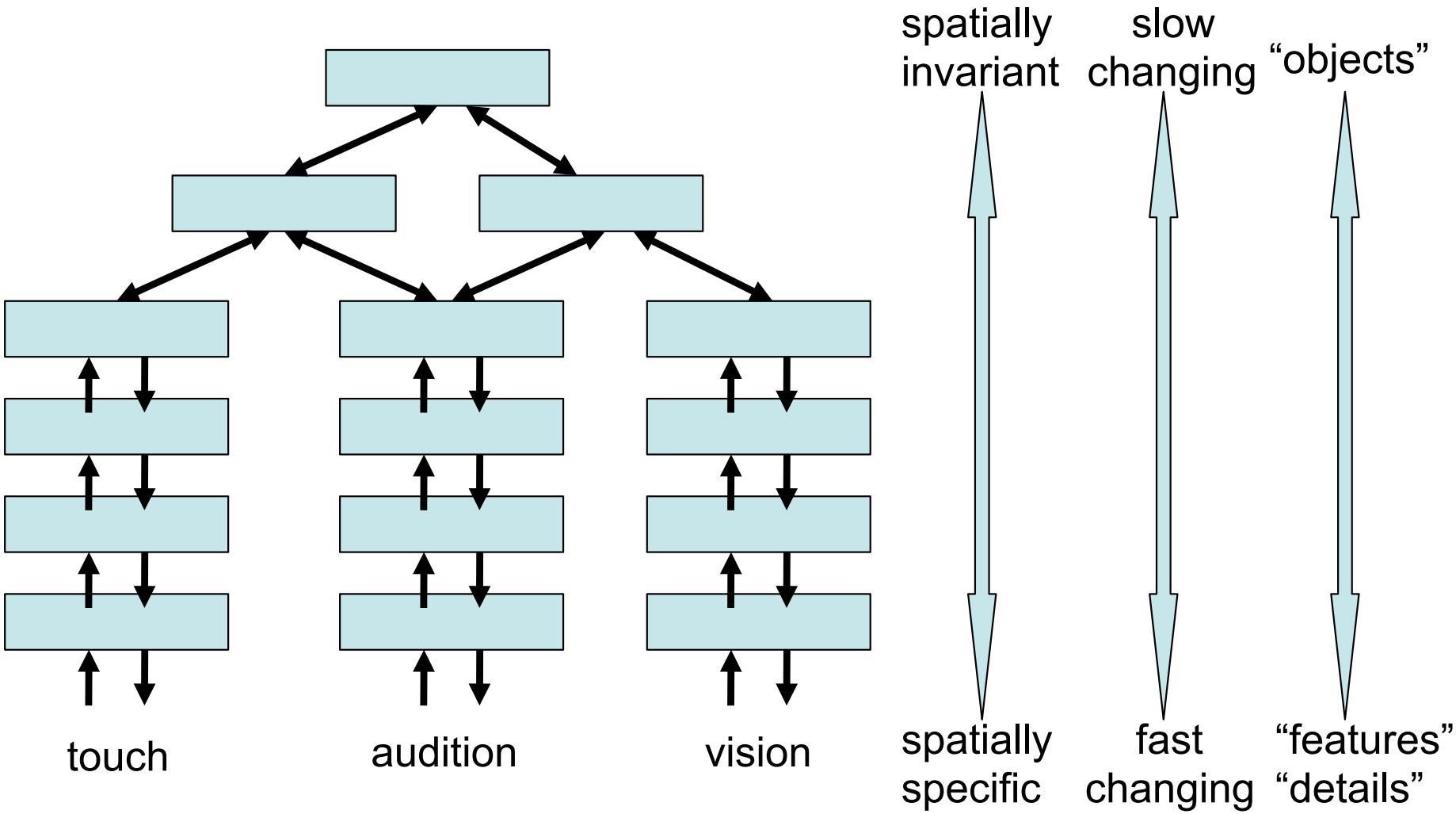


Invariant, corre xecution
identified and s
How we use th to us.
(we can also ic ning unexpected
is happening)



Hierarchical Structure Hypothesis

Reproduced from “On Intelligence”, Hawkins



Amodal Cortical Inspired Primitives for Structure

- Grammars: Clusters and Sequences → Sequences of Clusters hierarchically nested to form trees.
- A way of knowing what is happening at multiple scales in space and time. Thus inherently enabling prediction and verification. (Constantly predicting based on what you are observing now and in the past but know when you are going off the rails)
- Constant maintenance of 'Hierarchical Expectation' which are feed both up and down to improve prediction.

An example of the kind of approach we hope to explore

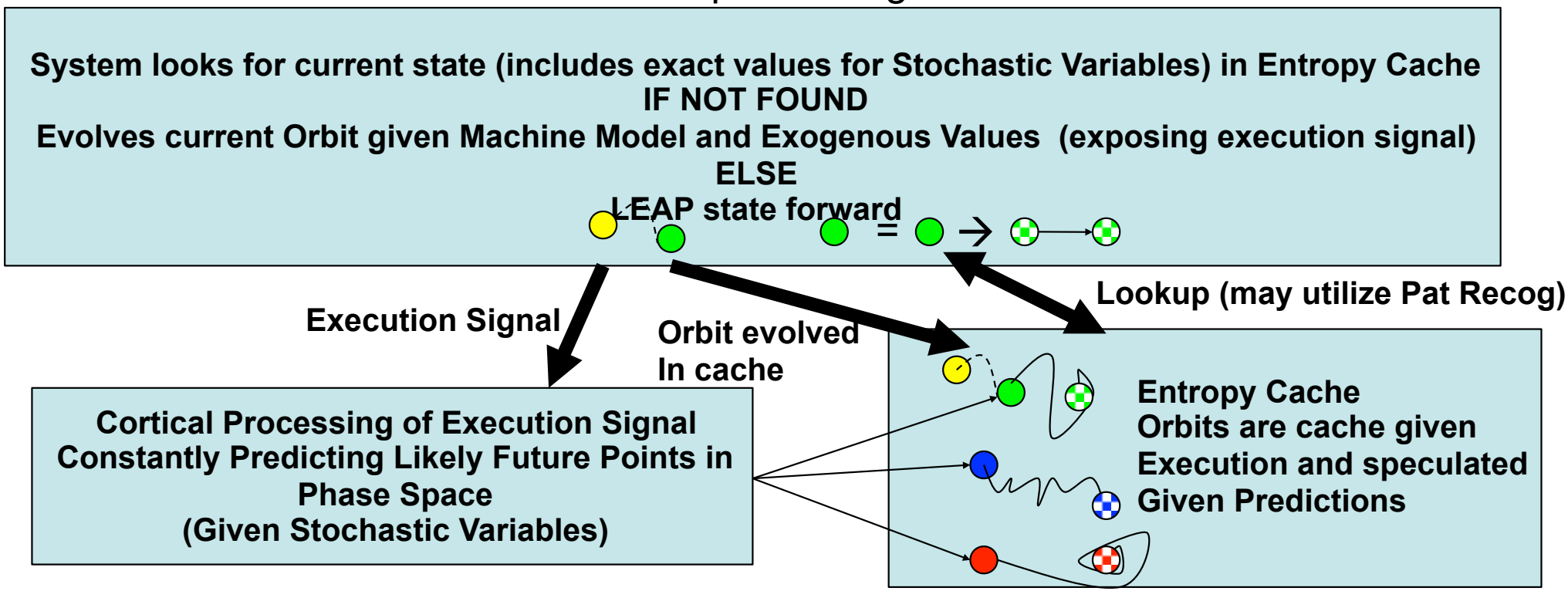
General Purpose Practical Deterministic Mechanisms as typified by Waterland's Dynamical Systems Simulator

+

Predictive ability of Cortical algorithms to exploit structure in generalized Stochastic signals (Assuming Structural Assumptions)

→

Hybrid which utilize a combination mixing stochastic processing and deterministic processing



Summary: A Different way of thinking about how to exploit more Transistors

- Rather than thinking about how to program all the transistors we want to exploit the majority to learn, store and extrapolate on the structure emergent in the Programmable Machine Model Execution. Not only accelerating the function specified (programmed) but enabling new unspecified function to arise.
- Interface/Program Statistical Learning mechanisms via a standard Programmable Machine Model.
- Auto-Parallelization not of the programs but of the underlying information transformation that the entire system achieves.

Implements **P**rogrammable **M**achine **M**odel Generating Execution

Learning, **S**toring and **E**xtrapolating on the emergent **S**tructure in the Execution of the Programmable Machine Model

Additional

Some Implications and Impacts

Some technical Implications:

- If execution can be converted to information then it should be possible to build a system whose storage capacity should have a direct relationships to its computational function. Eg the larger the storage the more capable. It turns out that this is exactly the property displayed by biological cortexes. Computational ability increases with pure increase in size by replication of the uniform structure of the cortex.
- Computation can be converted into search. Arbitrary mechanisms for search can be employed: Parallel, Quantum, Cortical, etc.
- Computation can be interfaced directly to machine learning.
- Machine learning can be applied via general purpose systems.
- Standard digital techniques can be used to program and interface neural based systems.

Scientific Implications:

- We perhaps closer at a understanding information processing and the relationship between logical and statistical processes.

Industrial Impacts:

- The potential to build new products and services which are fundamentally unique, proprietary and superior:
  - Cloud computing services in which a simulator takes traditional machines as input and is capable of exploiting high degrees parallelism to more efficiently execute them providing greater margins.
  - Smart machines which can be programmed in human friendly ways but exploit underlying structure.
  - Independence from ISA
  - Suggests new architectures which utilize the fact that such continuous optimization is taking place to simplify construction and programming burden
- Exploit the burgeoning field of neuro-computation.
- Establish a new kind of computer and field of hybrid computation in which MCS can be rigorously explored.

Fun with Structure

- behavesostrangely.mp3
- Dreams of Mice and Men :
  - Play it again Sam → Structure in Neural Spike Activity correlated by “Accident”
  - Dreaming of a day of Mazes and new Problems
- Ferrets seeing with there “Ears”.
- Feeling a rubber hand.
- Sensory equivalency : “tongue” sight

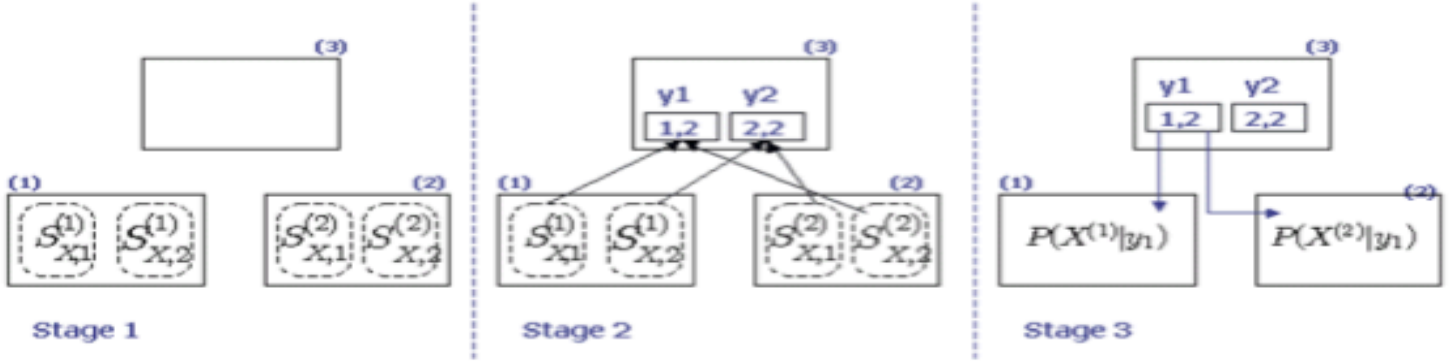


Fig. 1. Learning Stages: Learning starts at the bottom of the hierarchy and proceeds to the top. The modules at the very top of the hierarchy receive their inputs from a small section of the visual field. During Stage 1, these modules observe their inputs in time and learn the *most likely sequences* of a particular length of inputs. Once stage 1 learning is finished,these modules start passing up the index of the sequence whenever they observe one of the most likely sequences at its inputs. A higher level module gets its inputs from several lower level modules. During Stage 2, the higher level module learns frequent coincidences of sequence indices. These become the alphabets or concepts for the higher level. Note that this alphabet abstracts what patterns occur together in space and time. During the third stage of the learning procedure, the higher level concepts are fed down to the lower regions so that they learn the occurrences of the lower level patterns in the context of the higher level concepts. Repeating this in a hierarchy we obtain a graphical model as shown in figure 1

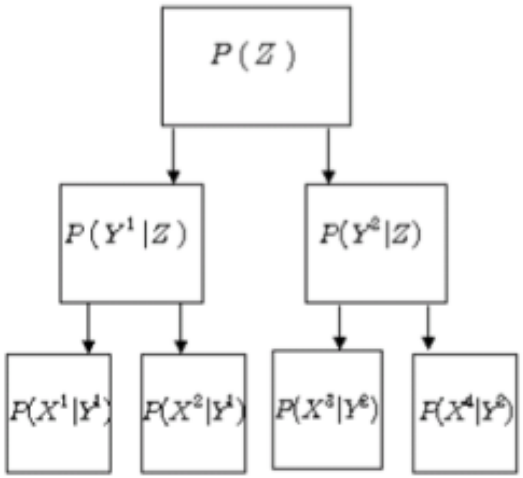


Fig. 2. Structure of a typical Bayesian Network obtained at the end of the learning procedure. The random variables at the bottom level nodes correspond to quantizations of input patterns. The random variables at intermediate levels represent object parts which move together persistently. The random variables at the top node correspond to objects. During training the definitions of the intermediate object-parts and then the top-level objects are obtained using algorithms described in figure 1. The probability tables are also filled according to Stage 3 of figure 1.

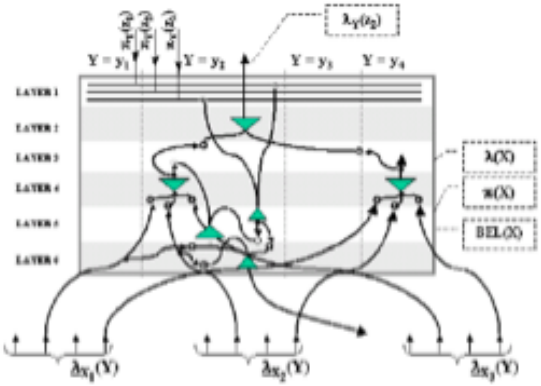


Fig. 3. Belief Propagation and Cortical Anatomy: The belief propagation equations that we used for inference in our model has an anatomical mapping which matches anatomical data [18] to a large extent. Shown here is the cortical circuit resulting from such a mapping. This mapping enabled us to replicate some of the physiological experiments in our system.

Better Slide on Clustering, Sequences, Prediction, Time, Global View/Context (All Boundaries Eliminated → Exposing structural relationship at all scales but unified)

What do we really mean by an Entropy Cache

- A Table and lookup methodology
- A Compression tool
- Search mechanism
- Memory mechanism
- A general purpose cache which can be used by dynamic simulation/optimization.
- A synonym for a component of a hybrid system which stores and retrieves information.
- A mechanism for storing and retrieving statistical structure.

Why should a Systems Software Researcher have anything to say about MCS

Intuition and statistics about execution are our bread and butter:

- Patterns and Probabilities in execution
- Patterns and Probabilities in memory access and data values
- Privy to all software layers and hardware structure.
- Caching and lookups are fundamental in our thought processes
- Confluence of global dynamic and static structure
- Global view and Exogenous Event Management
- Serious study of general purpose scalable software layer, hence forced to consider implications of large scale parallelism.

Jonathan's Background

Pre 2002: Parallel Computer Systems and OS research: Masslin, Gamsa, Smith, Liedtke, Rosenblum, Mazieres (LBFS), Tridgell (rsync), Peterson (Scout), etc.

2002 : Confluence of my research, ML milieu, and robotics hobby

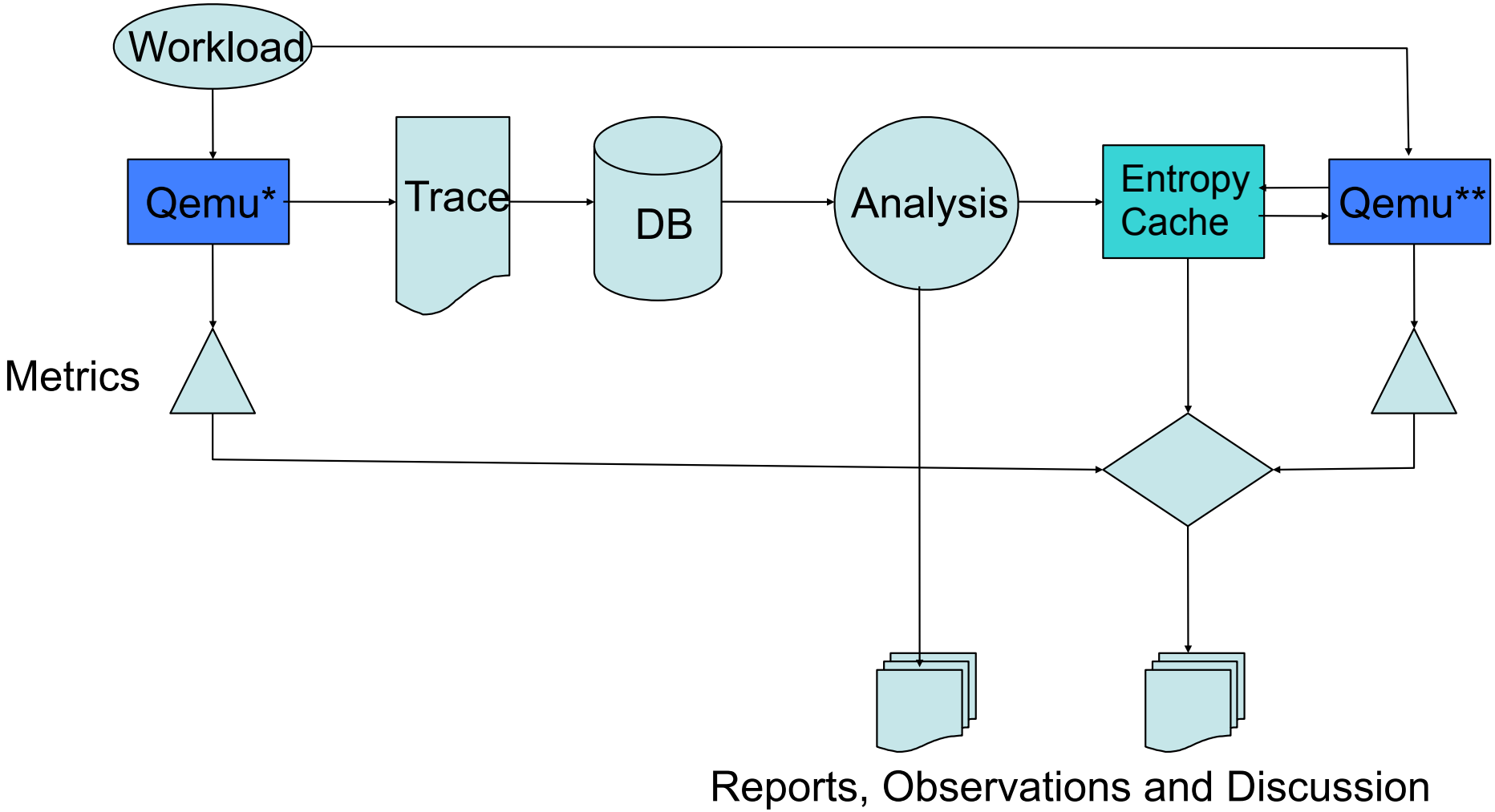
1) Interaction with Vision and ML students at Toronto (course *Decision Making Under Uncertainty*)
2) Dynamic Simulator
3) Self-optimizing System using Hypervisor based introspection, indirection, and modification

2003 – 2007 : Turing, Godel, Von-Neumann, Shannon, Bayes, Pearl, Granger, Dean, Mumford, Grenader, Hawkins, Mountcastle, Widrow, etc.

Looking for something more than intuition to guide construction. Want to characterize the phenomena first. Then collaborate and utilize the right techniques. Eg. Biologically motivated Bayesian inference is likely to be far more robust than adhoc voting schemes. I do not want to reinvent Pattern Theory or Neurology rather looking for viable integration framework.

Amos’s Background

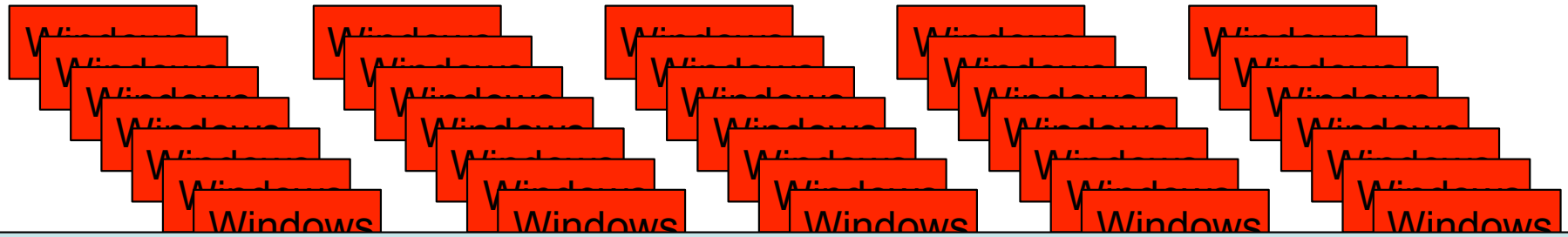
General QEMU Experimental Methodology



Project Plan: Initial Experiments and Hypotheses

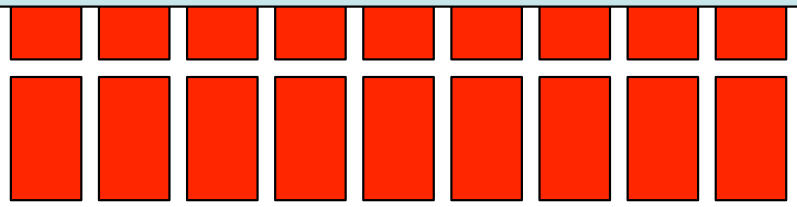
Workload	Parameters		
Async Responder	Response function	In,Out chan	Domain config & Lang
Webserver	Content	# con	Domain config
File corpus search	Content	# files	Domain config
Compilation	Source	Size	complexity
Mixed			

Running 1000s of Machines as a Single Workload



Virtualization to the Extreme

1. A virtual machine is faster than a “real” one.
2. Virtualizing 1000’s is more efficient than 1.
3. ISA independent optimization.



Exploit Structure at Large Scale

Some Questions to Keep in Mind

The 2000's version of the RISC questions -- questions of reuse, caching and compressing ***computation***:

- Is there redundancy/structure in the instructions executed?
- Is there redundancy/structure in data values?
- Is there redundancy/structure in Exogenous Events?
- Is there redundancy/structure in the joint probabilities?
- Is redundancy/structure in the *systems overall* behavior detectable?

Can structure in computation be exploited in generic ways to more "efficiently" implement a complete computational system independent of how it is programmed?

Dynamically accelerate what needs acceleration.

Automating the time space tradeoff and potentially exploiting alternate information processing models.

Exploiting structure across multiple scales and domains while leveraging redundancy.

Historical Slides of things that I have considered along the way

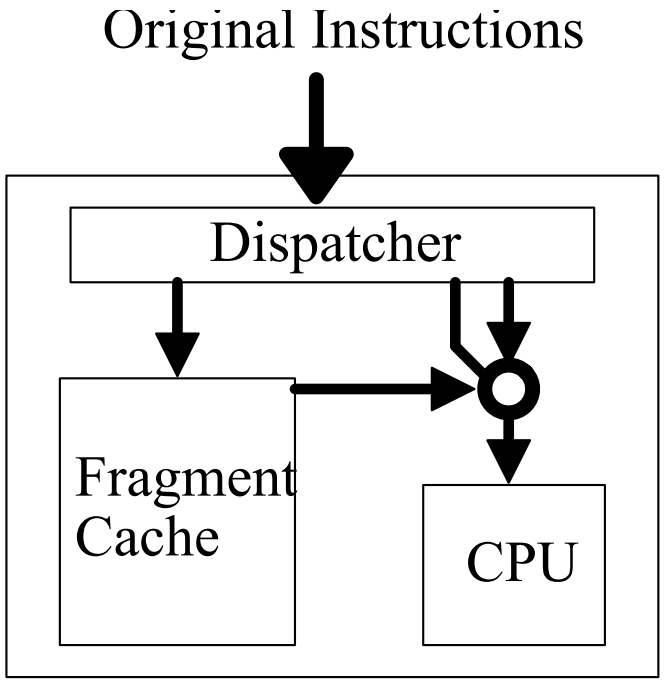
Complete System Optimization

- Can trace construction work across protection boundaries? For that matter arbitrary communication boundaries?
- Can traces/streams/fragments be first class primitives manage by the system?
- What are the equivalence of trace heads, trace exits, dynamo context switches etc when considering the entire system?
- Are deeper optimizations possible when all of the system behavior is being considered?
- What is the equivalent to the Dynamo loop assumption when general purpose system are considered?

System Trace Cache Support

With a little though any of us can come up with several architectures and technical solutions for system wide trace cache integration. The question however, is not how but why. Before we move on to why let's walk trough a couple of possible architectures.

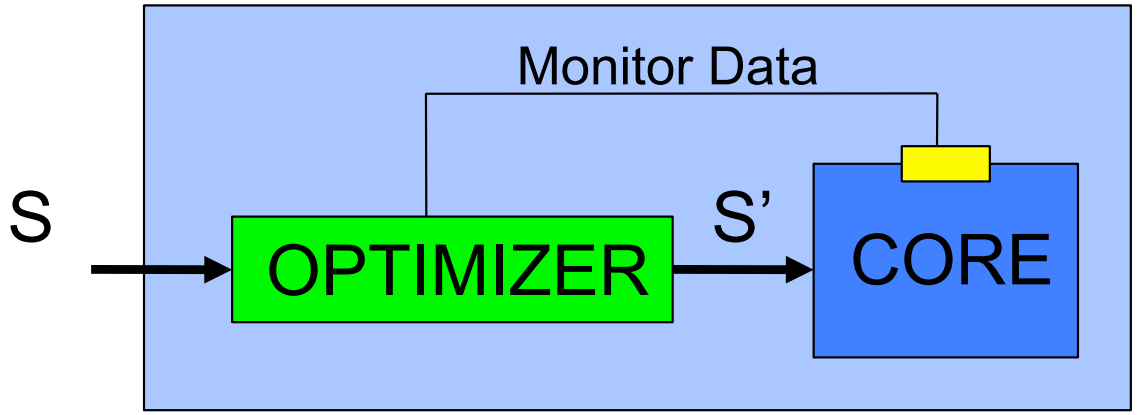
1) Straight forward integration of Dynamo like approach
2) More aggressive Parallel simulation approach



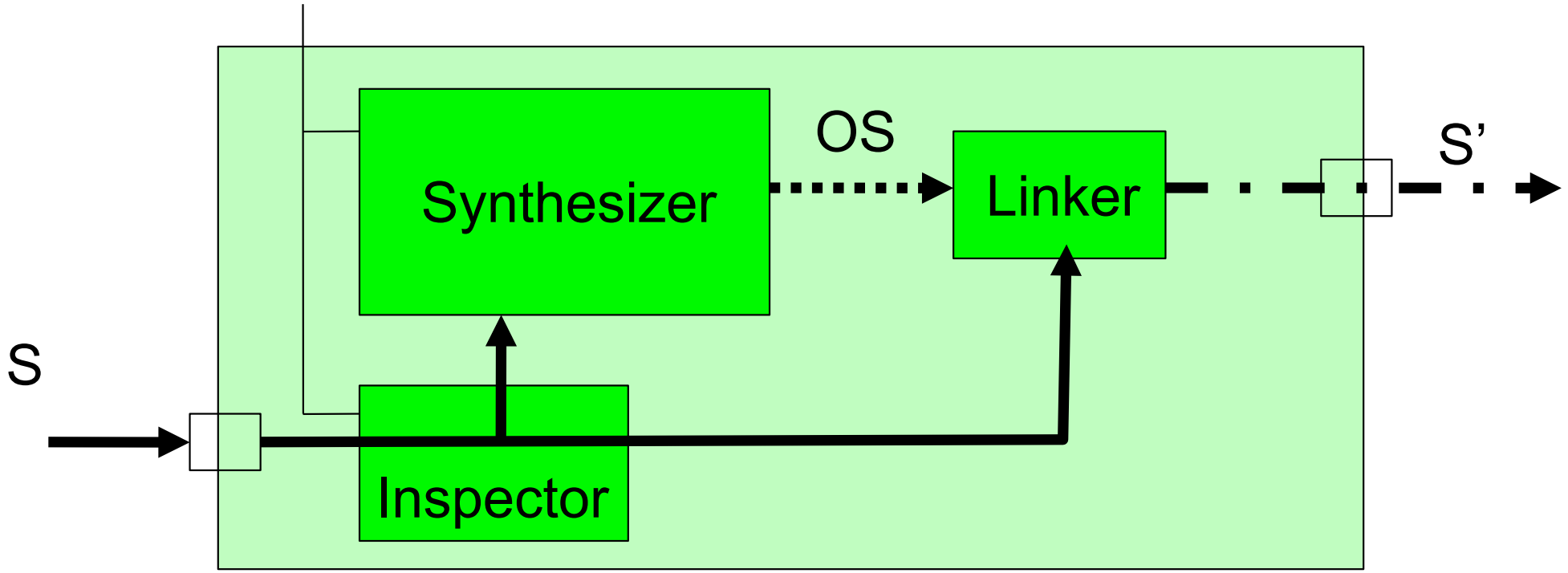
System Trace Caching – One Possibility : Building Virtual

S : Instruction stream in ISA of virtual CPU as generated by complete SRO
S' : Instruction stream presented to real CORE
OS: Optimized instructions for CORE

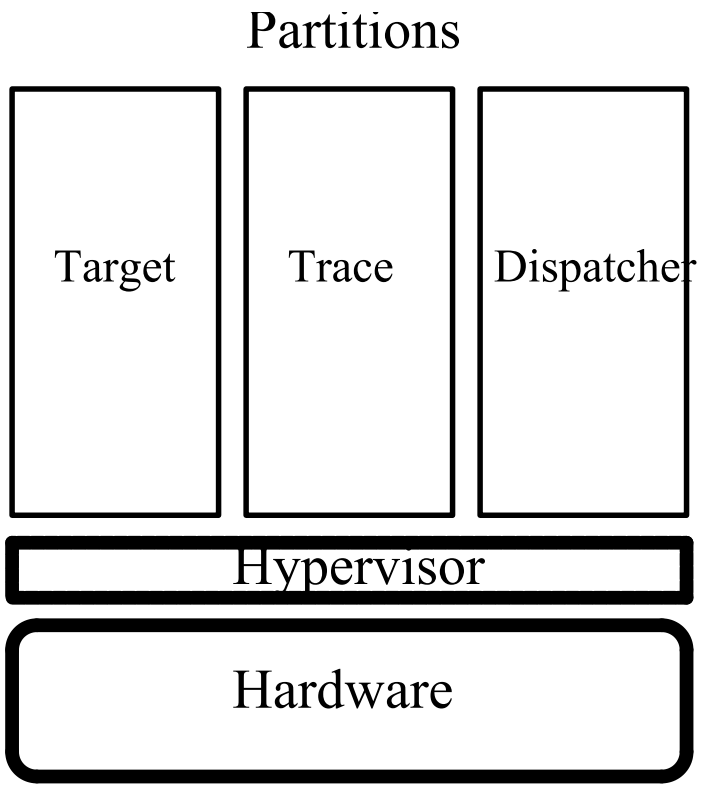
$$S' = a*S + b*OS$$



A Dynamic Optimizer Component with Global Scope and Protection



A Hypervisor Realization

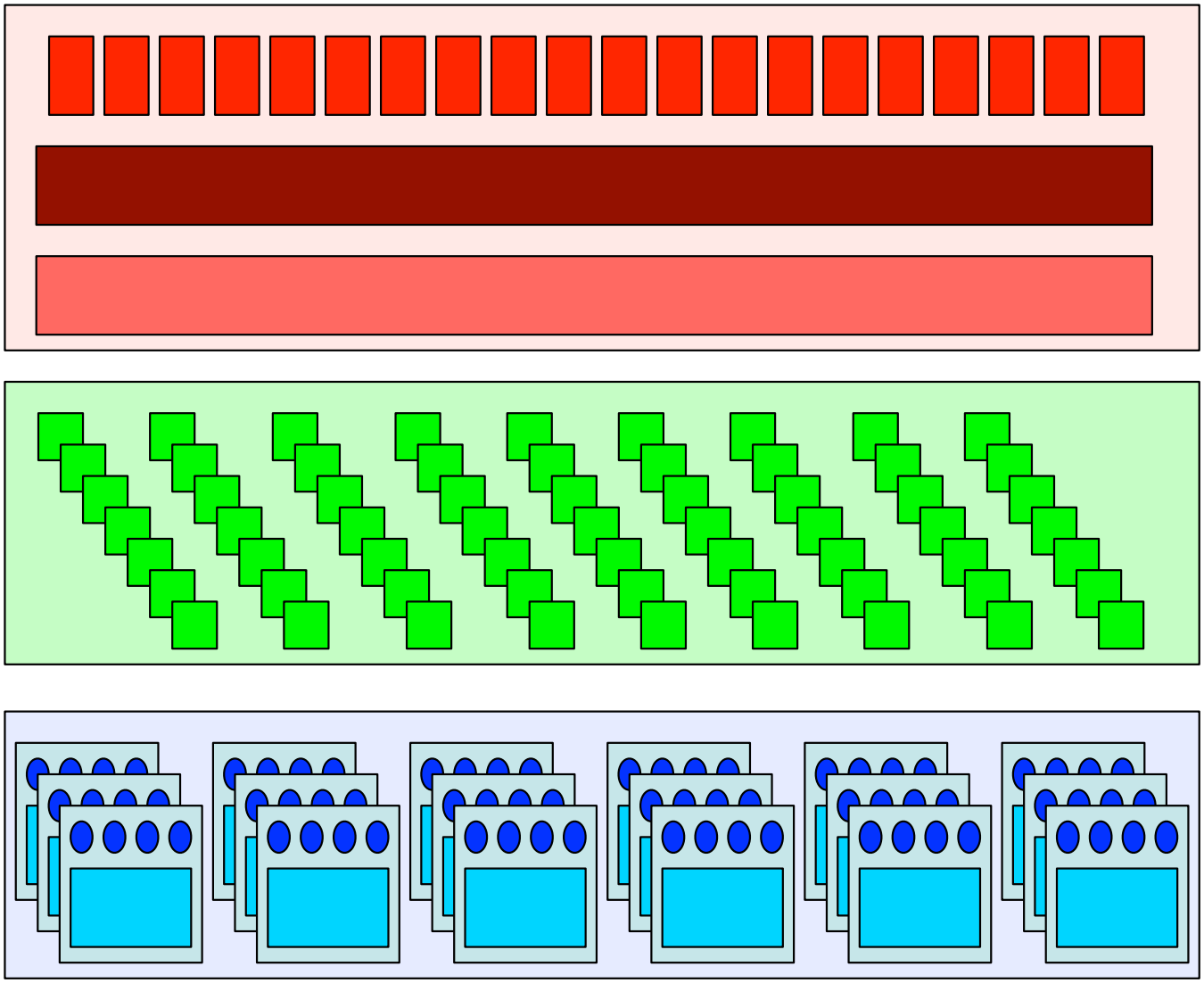


1. How to start optimizing
   - Introspection on exogenous events → hypervisor allows Dispatcher to register introspection functions on interrupts to target partition
   - Hypervisor exports Hardware Performance monitoring control to Dispatcher
2. Control transfer to trace :
   - Place Partition Branch commands (IPC) at trace entry points control transfer from trace
3. Some Mechanisms required:
   - Shadow Partitions : A partition which has access to all of another partitions memory in place. Plus its own memory protected from the Casting Partition.
   - Branch to Shadow Partition (IPC): Causes transfer to shadow
   - Step Partition
   - Read Partition
   - Write Partition

Computation as Simulation

1. All computation can finally be realized by a simple machine (UTM).
2. HW and Software have (d)evolved to implement various forms of (de-)virtualization which are primarily to support software abstractions.
3. Significant portions of computation can be cached.
4. The larger the time frame and the great the view the cache can have the more effective it can be.
5. Simulating 1000 of machines is more efficient that simulating 1

Go big or Go Home ;-)



VM instances

inspector

dispatcher

global cache
HW & Software

unified
underlying
HW

Things we have considered doing

(a) system wide generalized trace cache, and

(b) a system wide generalized infrastructure for trace management, manipulation and transformation

(c) identification/marking of "Interesting" event :

- i. External Interrupts
- ii. Loads and Stores to "meaning full" addresses
- iii. Synchronization primitives
- iv. Shared Data access

(d) Unified support for hardware performance monitoring with respect to partition optimization

(e) A framework for using -- Qemu and Hercules to study complete machine execution across multiple ISA's and OSes –

- i. Target Partition Creation: Launching Qemu and Hercules in "raw" partitions on top of either HW, Xen or Rhype on either PPC or X86-64 depending on resources and availability -- here we would use either a hand configured minimal Linux kernel, the K42 microkernel, L4 or LibOS as most appropriate to act as a base run-time to run one of the above in a partition.
- ii. Extend the hypervisor as necessary with mechanisms to support introspection and manipulation of a Target Partition including:
  - A. Suspension
  - B. Single stepping
  - C. Tracing
  - D. Modification
  - E. Profiling
  - F. Marking
- iii. Construction of analysis techniques and infrastructure which allow hypotheses about structure to be tested. At this point we are not necessarily concerned with specific optimization techniques but rather to see if certain types

Back to Why

- How to implement system wide caching is not as important as understanding what there is to optimize → must establish that it is worth it.
- As such we are focusing on studying computational phenomena at large scales. Both within a traditional SRO instance and across instances.
- We want to be able to design the infrastructure based on real data :
  - Are traces really the right thing to cache?
  - Can arbitrary boundaries and sequences of unfolded deterministic code be found and eliminated?
  - Is it worth dedicating hw resources either in the form of caches, state machines or morphs to accelerate the components?
- Scientifically worth exploring relationship between Computation and Information. Can general computation be expressed as search/compression. UTM as Information generator.