

Large-Scale Modeling of Epileptic Seizures

And its relationship with the analysis
of electrode arrays

What is epilepsy?

- Epilepsy is a Central Nervous System (CNS) disease characterized by large scale excessive or synchronous neuronal activity
- Its external symptoms include:
 - Loss of consciousness
 - Jerking movement

Why does society care?

- Possibly hundreds of Millions of people are affected worldwide (1-2%)
- About 1/3 of patients (millions) do not respond to medications currently available
- Medications have significant side effects
- Difficulties in treatment may be attributed to our uncertainty of what causes epileptic seizures

Why Modeling Epilepsy

- One unresolved issue is how individual neurons' behavior affects large neuronal networks (and vice versa)
 - No currently available experimental technique can provide that information
- Testing large sets of parameters would be difficult or impractical in vitro or in vivo
 - simulations are necessary to guide experiments
- Models can be used to test inconsistencies between large scale and small scale neural tissue models and to validate statistical inference tools

HPC resource are necessary

- The human brain has tens of billions of neurons
- Its network is made of trillions of connections
- People and their brains are very variable in a multidimensional way:
 - where does normality finish and disease start?
 - How many “configurations” do we need to model?
- many alternatives need to be ruled out

What is in our model

- A LOCAL Neuron model:
 - Receives input (just the signal, not from whom or where)
 - Reacts to the input and its own status
 - Sends output to the network
- A GLOBAL Network model:
 - List of neurons connected to each neuron
 - How are these connections (speed, intensity...)
 - Possible changes in connections

Neurons

- Neuron type:
 - Pyramidal (excitatory -> excite other neurons when active)
 - Basket (inhibitory -> inhibit other neurons when active)
 - Chandelier, Double Bouquet, Stellated ... Potentially even thousands of types
- Neurons receive input until they spike:
 - on/off behavior: no spike, no signal to other neurons
 - Except for gap-junctions...
- We model neurons for their electrical not biochemical properties

Neuron models

- Switches: if enough spikes are received, spike
- Leaky switches: if enough spikes are received before their “total” leaks out, spike
- Segments, e.g.:
 - Dendrites (tree that collects most of the input)
 - Soma (main cell body, might receive some input)
 - Initial segment of the axon (start of output, might receive input)
 - Voltage gated channels follow specific PDEs

Problem with programs that simulate Neuron models

- It is very difficult to scope the software project:
 - We don't really know what causes epilepsy
 - We know what should be in the model and neurons can be fiendishly complicated cells
 - Models need to be very flexible to carefully test alternative hypothesis and find “crucial” experiments to determine which model is “true”:
 - Ability to modify some characteristics, leaving others unchanged
 - Ability to use different neurons on the same network and different networks with same neurons
 - ...
- Communication is random both in space and time
- Performance and flexibility tend to be mutually exclusive and both require a lot of work.

Networks

- Network carries the signals generated by the neurons connected to it
- Signals will take from
 - Nearly instantaneous (gap-junctions)
 - The slowest connection
- Both local (most connections) and long distance connections play a role in creating network signals

Simulation sizes

- Smallest:
 - Epilepsy is often observed with EEGs, i.e., electric signals measured by macroscale electrodes
 - Between 1 and 10 million neurons with 100 million to a billion connections without counting the “rest”
 - Largest:
 - CNS contains 10s of billions of neurons and 10s of trillions of connections
- The human genomes has only 3 billion base pairs

Basic implementation: Neurons

- Neurons are objects belonging to classes (simpleNeuron, segmentedNeuron,...)
- Classes have inheritable and polymorphic methods: creators, destructors, get (status), evolve (in time), modify
- Neurons don't know on what node they are, where they are in space, but have ids and clocks (to move asynchronously)

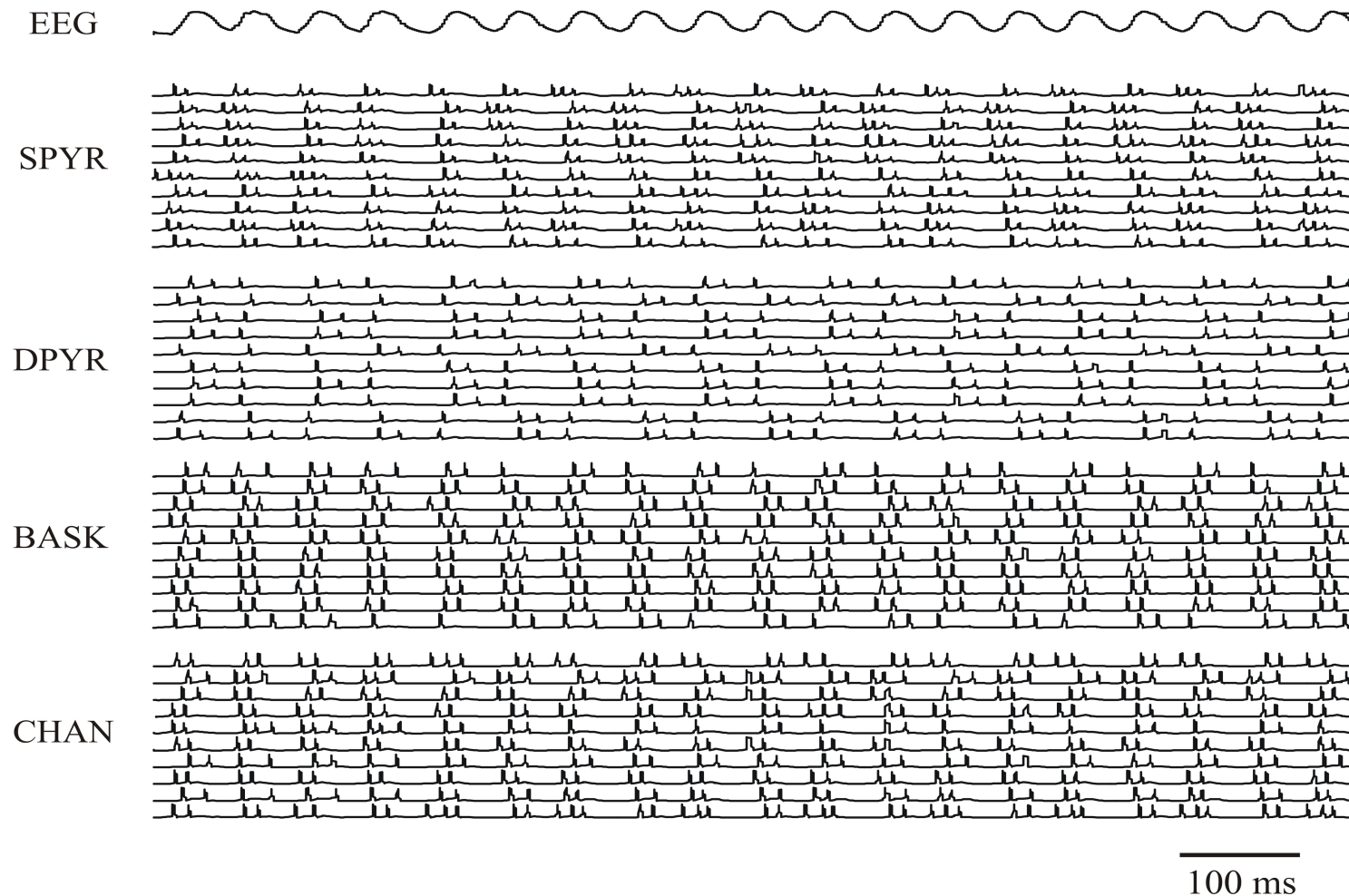
Basic implementation: Network

- Each PE (as in MPI rank) gets a part of space or brain
- The PE, via OpenMP, puts neurons on that space and in an array. Characteristics can be randomized
- Each PE broadcasts its neuron locations and characteristics to the other PEs
- Network is created using random numbers and probability of connection
- The network is stored on the receiving PE

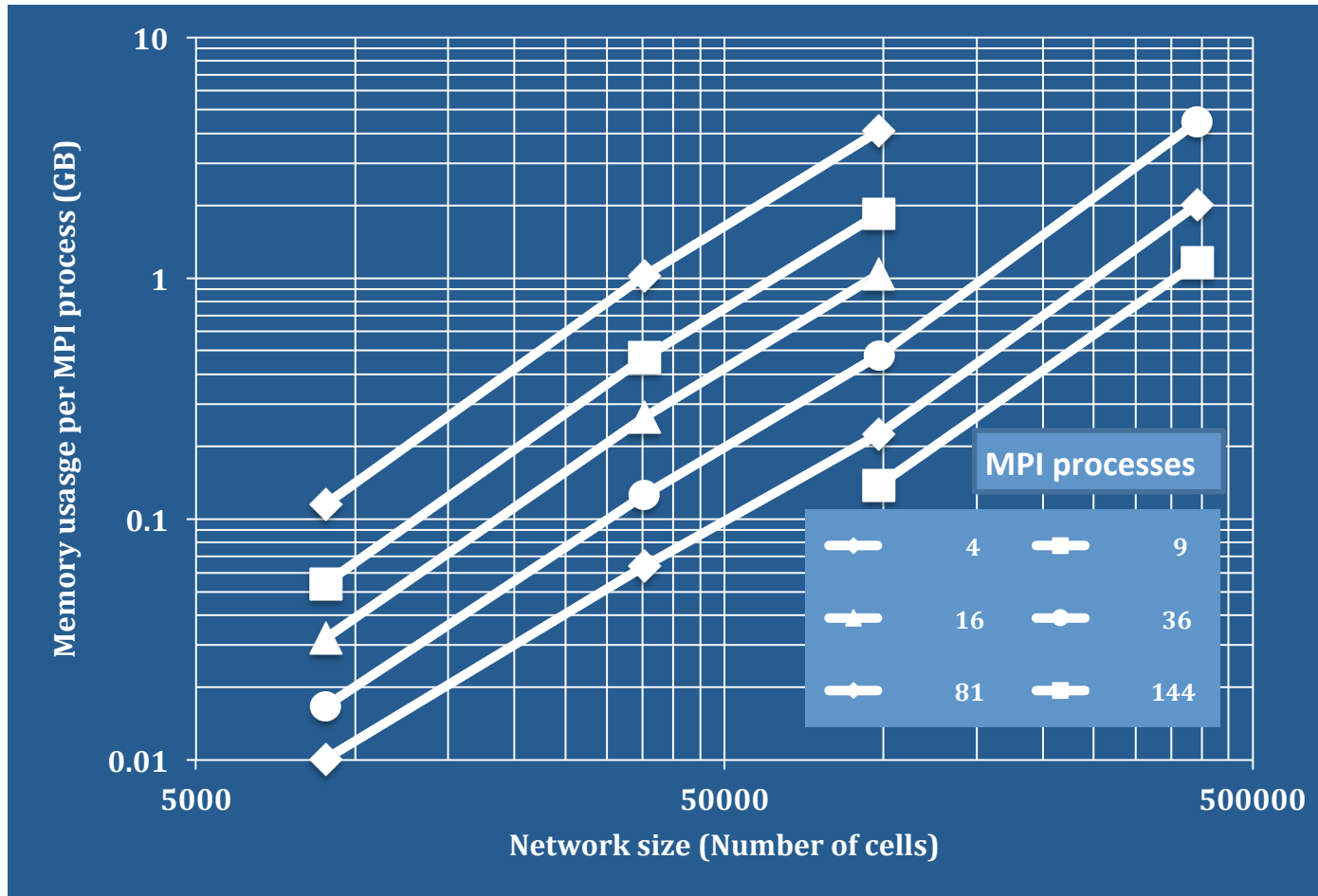
Basic implementation: Dynamics

- Neurons spike
- PE broadcasts its spiking neurons
 - broadcast is used because it is nearly always true that a PE will have a neuron affected by one of the spiking neurons of any other PE
- PE receives spikes, and puts them in a queue
 - because the Interconnect delivers the spike at a different speed than the physical model
- Checkpointing and I/O are done asynchronously by PE

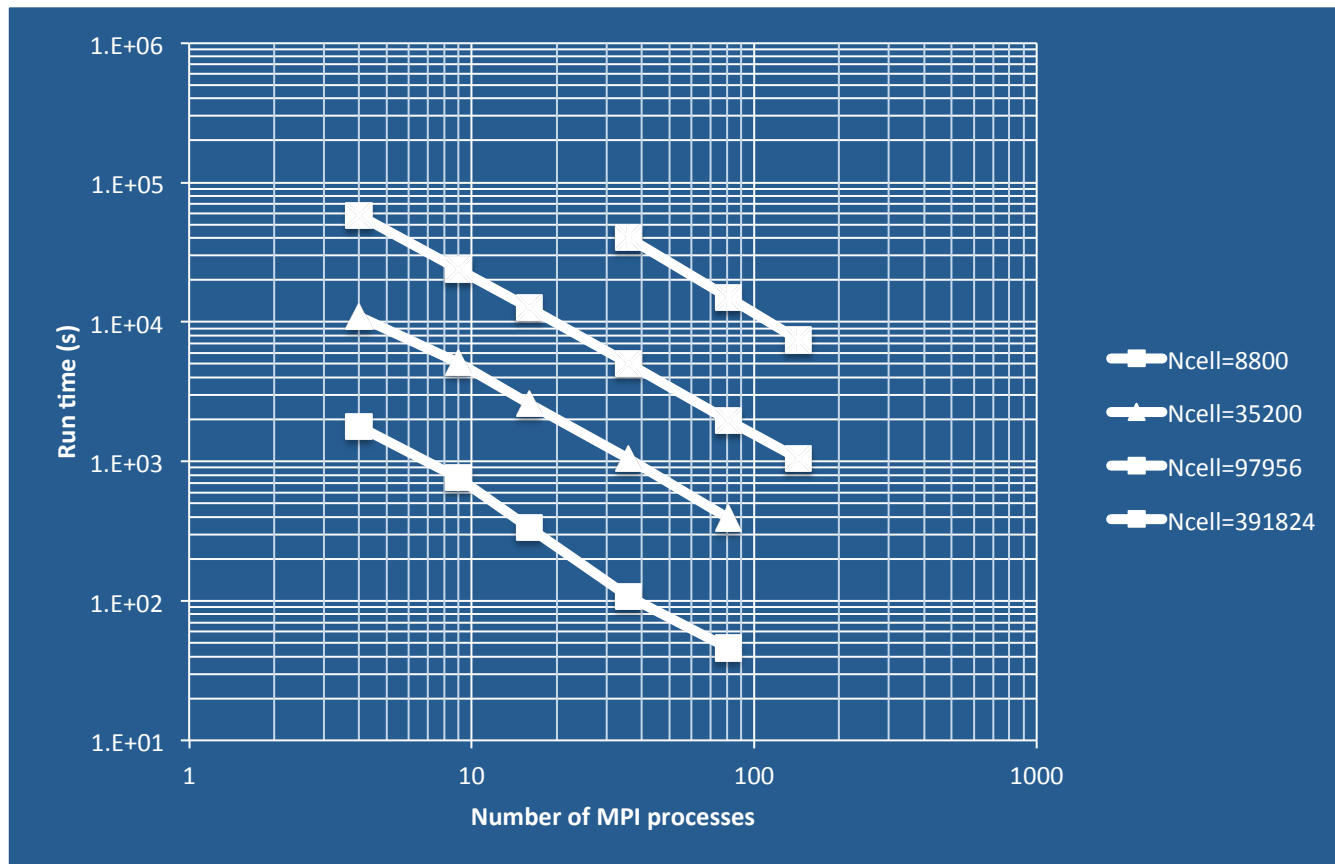
Emergent epileptiform activity



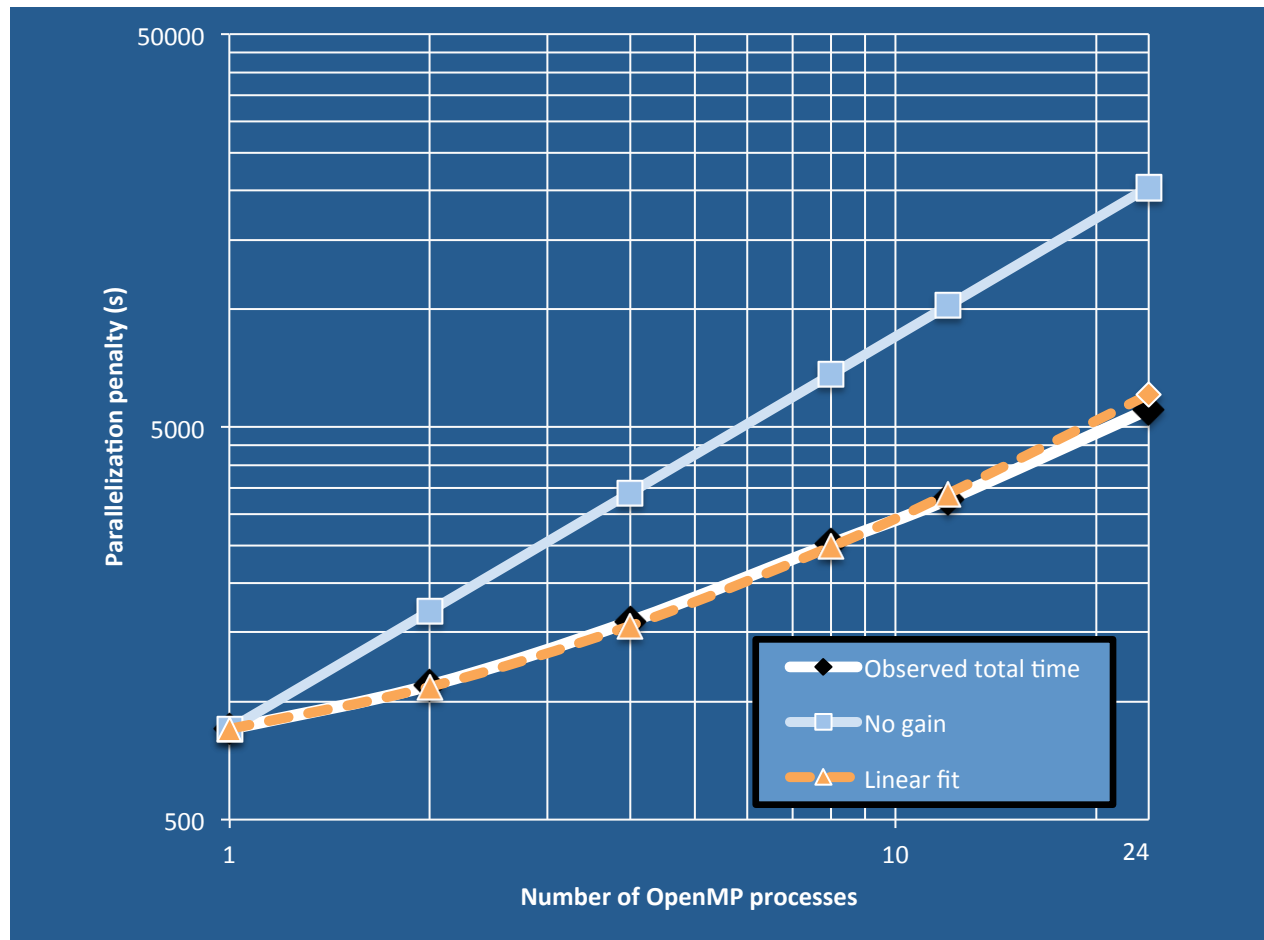
Memory scaling



Strong scaling: MPI



Strong scaling: OpenMP



Percent time spent on barriers (synchronous simulation)

Network size	Number of MPI processes used			
	9	16	36	81
2.6K	34%	42%	56%	77%
32K	25%	27%	34%	43%
94K	35%	38%	42%	46%
380K	44%	48%	53%	56%

Weak scaling (nodes \sim # neurons)

Network size	Estimated size of MPI pool	Expected computation time(s)
8.8K	4	1.8K
35K	16	2.6K
98K	45	4.1K
390K	178	6.8K

Number of synapses per neuron still increasing

Future developments: Asynchronous simulation

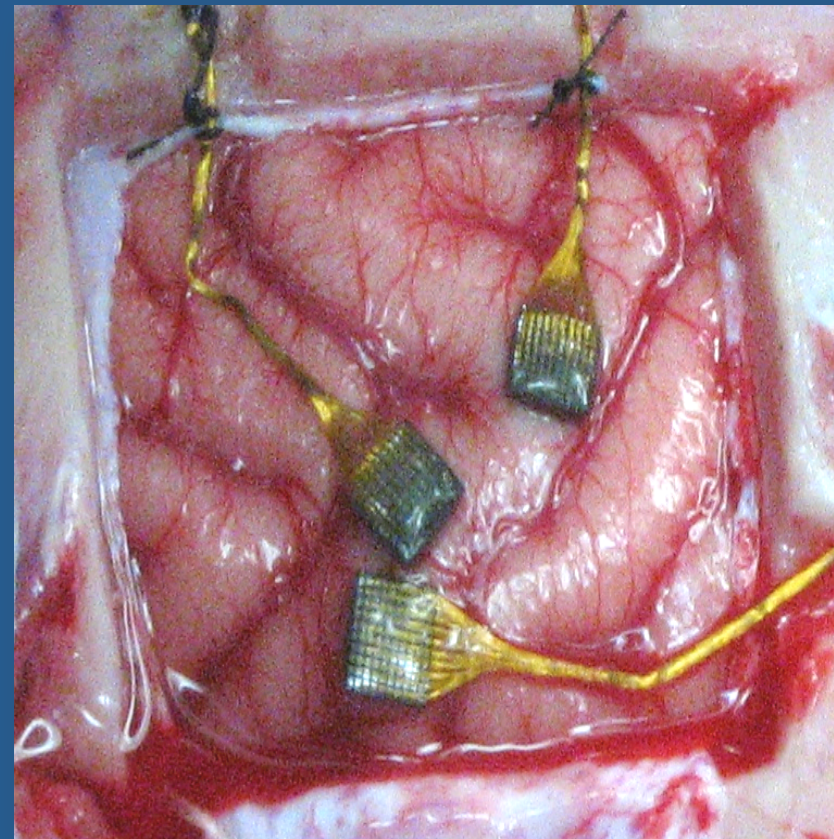
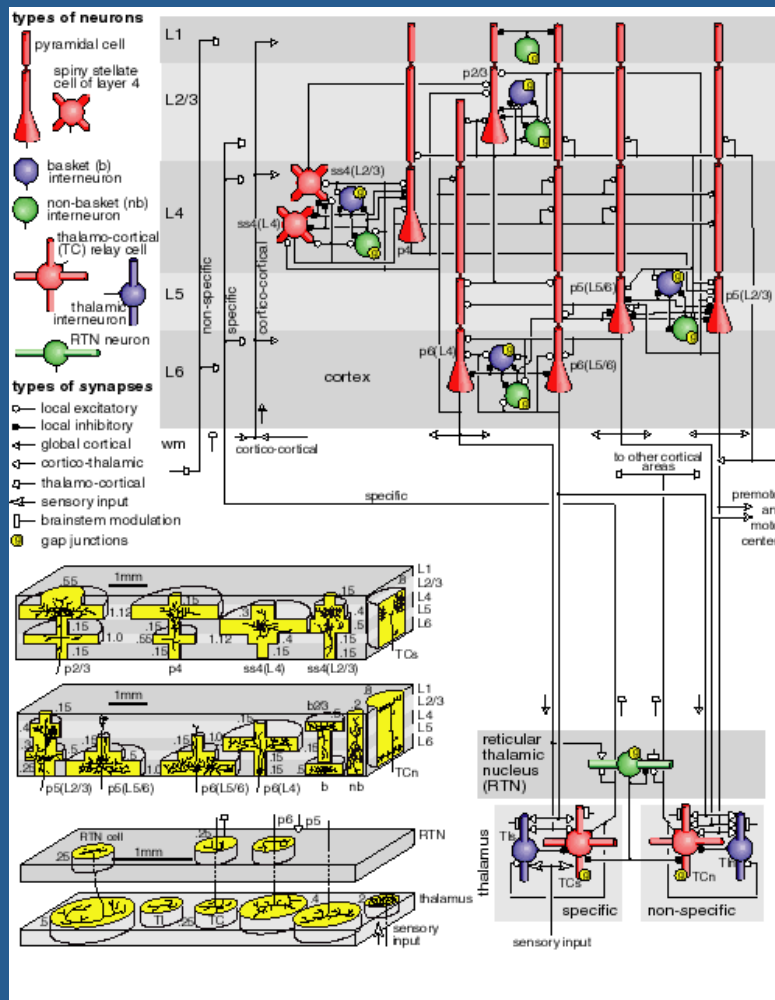
- Based on 'Integrative Model for Parallelism' *
- Short range / long range modeled by sparse matrices with different bandwidths: use fractional time steps to model delays
- No explicit barriers: synchronization only through task dependencies
- Combination MPI + OpenMP, overdecomposition
- Work in progress.

*Eijkhout, V., "A DSL for Integrative Parallel Programming," Parallel and Distributed Computing (ISPDC), 2014 IEEE 13th International Symposium on , vol., no., pp.27,34, 24-27 June 2014

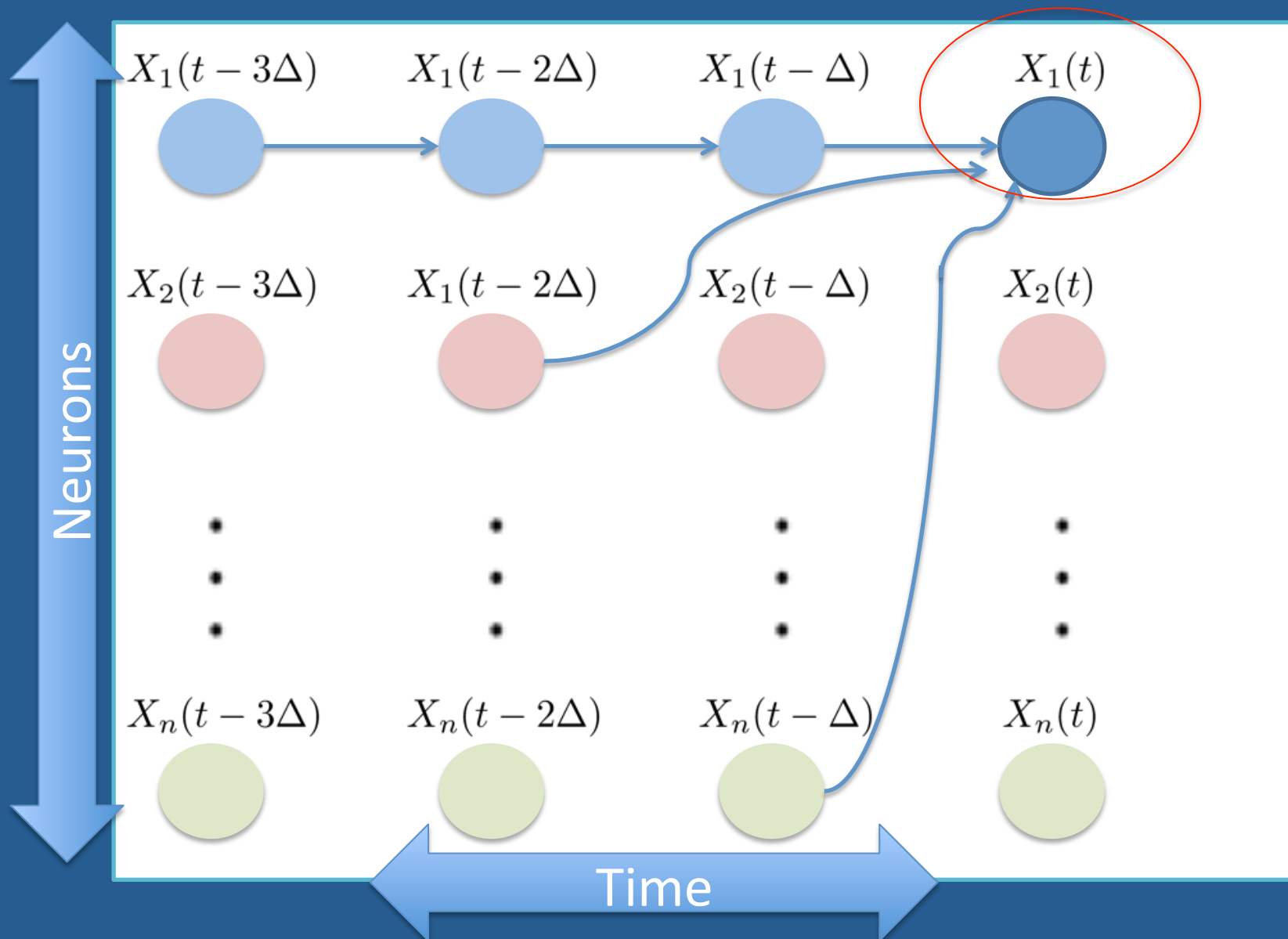
Conclusions about epilepsy models

- Simulate a complex dynamics of large collections of neurons to predict a relatively simple emergent experimental behavior
- Current limitations: flexibility, threading parallelism, need to reduce synchronization

Linking between simulation models and real brain physiology



Network of causally related neurons



Analysis of experimental arrays

- Latest recording technologies enable us to record spiking activities from tens to a few hundreds neurons
- Estimate causality: which neurons fires, directly or indirectly into other neurons modulating their spiking behavior
- Two types of data that require supercomputer
 - Large number of neurons per day (Feeding)
 - Large number of longitudinal data sets (BMI)

Decomposition of the Computation

Necessary to make the problem tractable:

- Analyze different behaviors independently
- Break time windows relevant to the analysis and analyze independently
- Model each neuron independently
- Separating all the time dependency lengths terms
 - one model for each time depth, then select with AIC/BIC

Error estimation

- There is no good error estimation model:
use resampling scheme
- A (smallish) 100 sample Bootstrap set of a
(medium sized) 30 neuron set with 15 time
windows requires approximately 5 million
program runs

Design of computation

- Swift* Workflow manages millions of program runs on the Cray XE6 nodes
- Programs on nodes are run in blocks by the *parallel* utility
- Each single program run is designed with a clear “interface” so that compiled Matlab, c/c++ or R estimation steps can be used

*<http://swift-lang.org>

Workflow scripting language: Swift

- Simple c-like syntax
- Build dependencies as file dependencies (a program starts if its input files are ready, there is no need to make that dependence explicit, e.g., `after_ok`)
- Easily adaptable to different schedulers and machines (from a Cray XC to a Condor cluster)
- Easy to build wrapper functions to allow logically equivalent but computationally different steps to be tested/used

Computing with parallel

- Very simple to keep n threads working over $m > n$ independent tasks
- Easy to build even complex data movement from Lustre to RAMdisk independently from the actual computation while minimizing I/O into Lustre
- Reduces the need for Swift to control millions of tasks as we can reduce them into blocks of tasks (swift is very good at handling dependence, but it might become unstable)

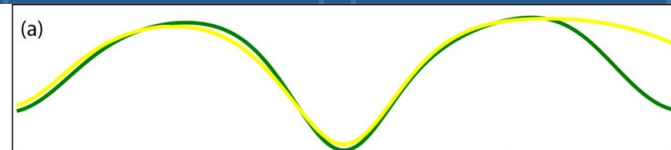
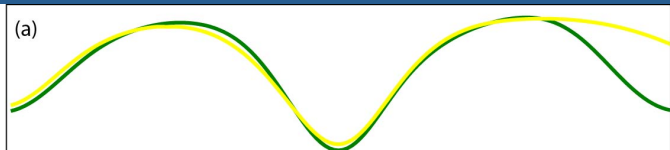
Single program tasks

- Allows us to test small model and then scale
- Allows to use many alternative statistical models that were not built for HPC without having to recode much
- Matlab: compiled for speed, avoiding GUI clutter, having to license millions of copies. Little recoding necessary
- C++ for speed, but reduces flexibility and usability by non programmers
- R: needs to be build for “packing”, but works fine also requires minimal coding to scale

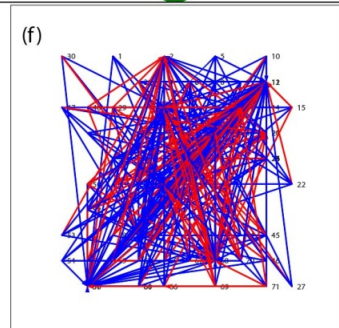
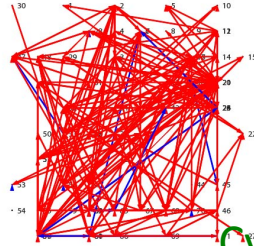
Granger Causal Networks

(Chew)

(Chew->Swallow)

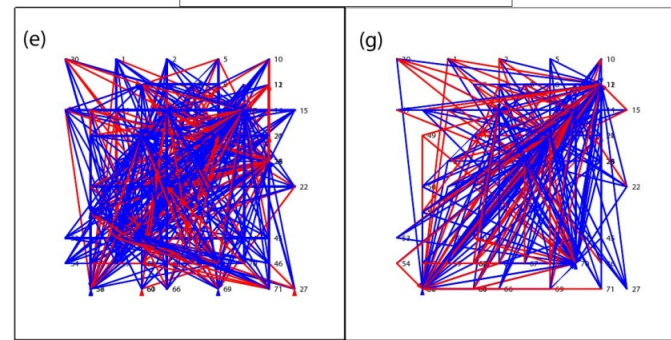
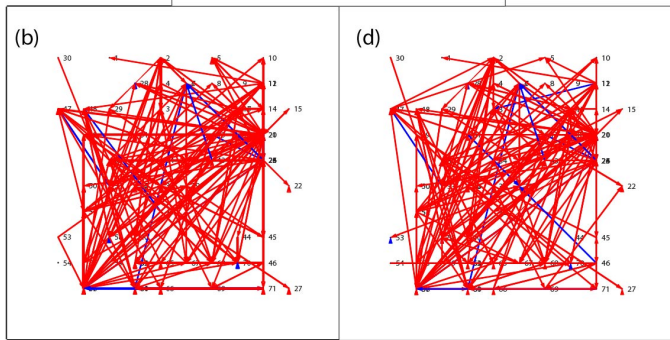


Cycle 1-2



Cycle 1

Cycle 2

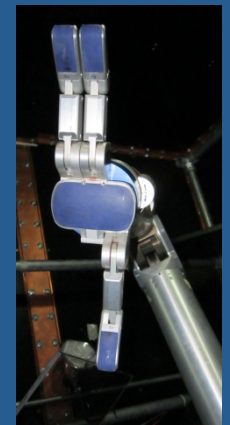
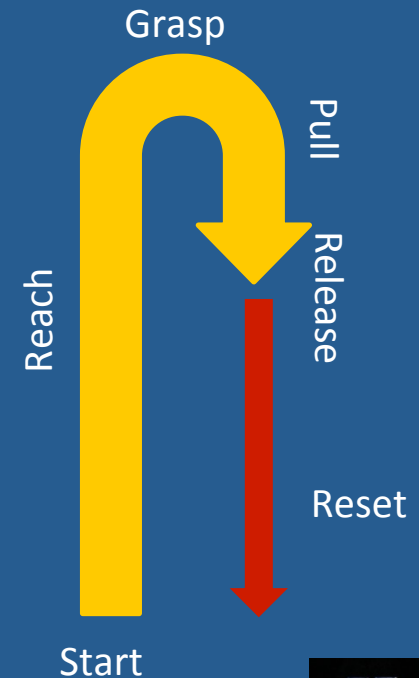


-300 ms -150 ms 0 ms 150 ms 300 ms

-300 ms -150 ms 0 ms 150 ms 300 ms

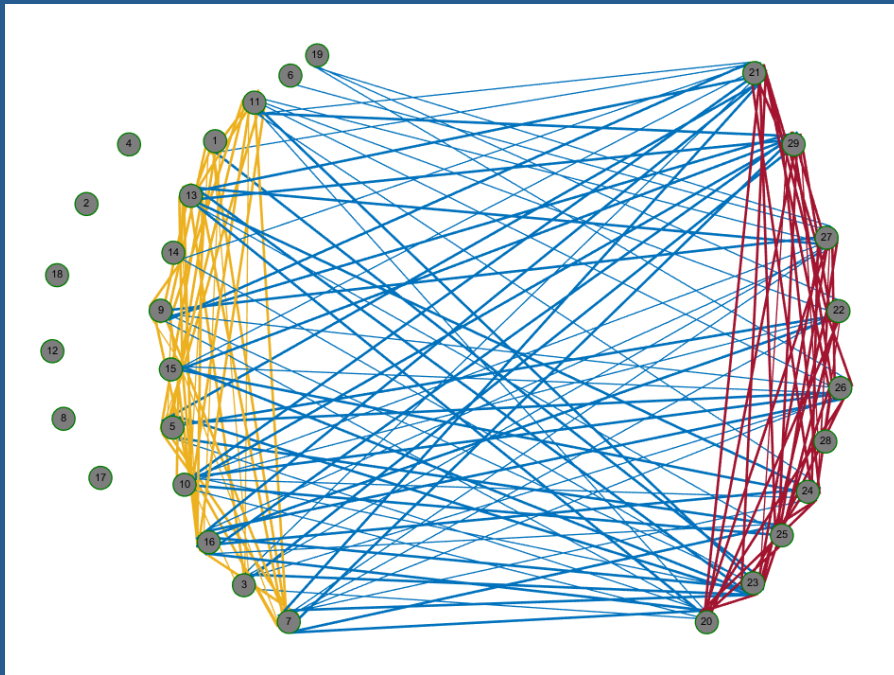
Brain machine interface (BMI)

- 39 days of recording
- 29 Neurons used for analysis
- Monkey trained to map her neural activities to control a complex robotic arm

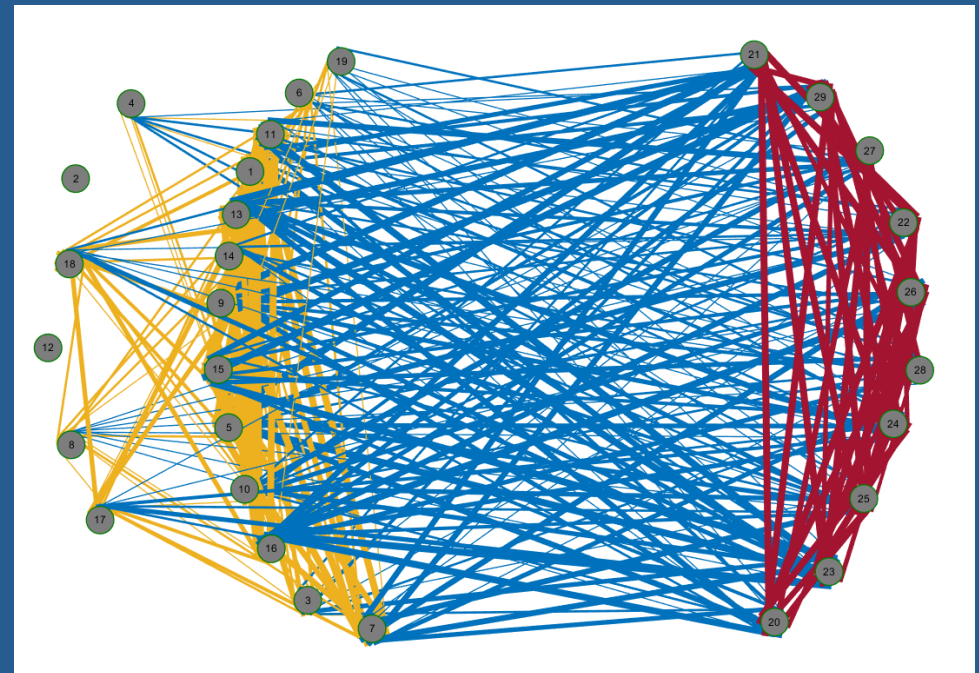


Topological change over BMI learning




Day-1 Cluster



Day-37 Cluster



Connection weight indicates stability

-  Within Reach Cluster
-  Within Grasp Cluster
-  Between the clusters