Parallel SNNAP: from object-oriented design to multithreading

Evyatar Av-Ron, Yidao Cai, John H. Byrne and Douglas A. Baxter
Department of Neurobiology and Anatomy, W.M. Keck Center for the Neurobiology of Learning and Memory
University of Texas-Houston Medical School, Houston TX 77030

The University of Texas Medical School at Houston
A part of The University of Texas Health Science Center at Houston

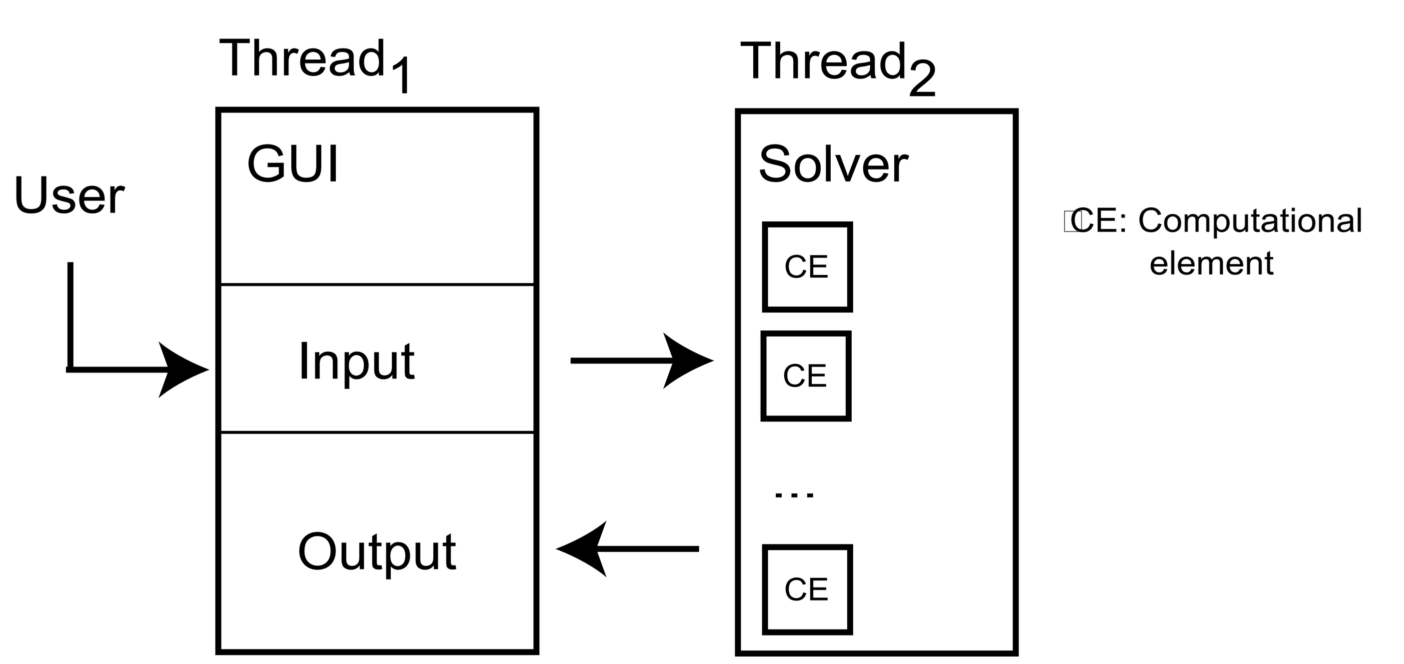
1032.7

Introduction to SNNAP, OO design and Java

SNNAP (Simulator for Neural Networks and Action Potentials; http://SNNAP.uth.tmc.edu) is a versatile and user-friendly tool for rapidly developing and simulating realistic models of neurons and neural networks. SNNAP is written in the Java programming language, and is portable to almost any computer. SNNAP is being redesigned to take advantage of OO features of Java (see accompanying abstract by Cai et al.). An OO design provides many benefits, such as a multithreaded architecture.

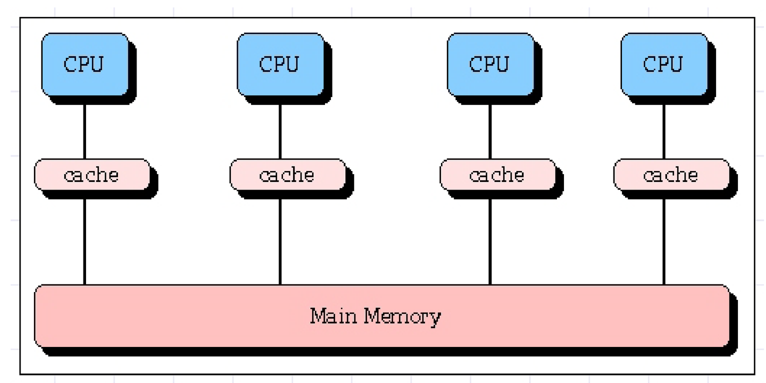
A multithreaded architecture provides an execution framework for parallel processing. Simulations often incorporate multiple 'computational elements', such as individual neurons in a neural network or individual compartments in a model neuron. In an OO design, each computational element is an object, and as such, may be executed on a separate thread. On a parallel computer, multiple threads can be processed in parallel.

An example of classic dual thread architecutue is shown below with the graphical user interface (GUI) running on one thread, and the model solver running on a second thread.

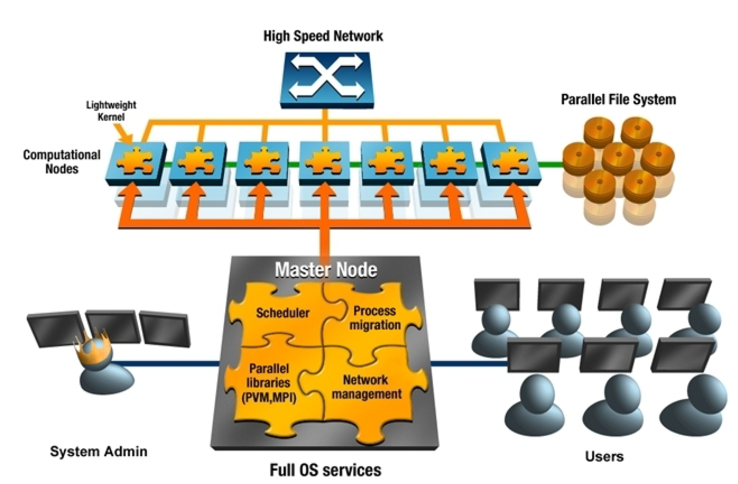


Parallel computer architecture

There are four major types of computer architectures that are used for parallel computation: i) parallel vector processors, ii) shared-memory, symmetrical multiprocessors (SMP), iii) distributed-memory, massively-parallel processors and iv) distributed-memory, cluster computers (often referred to as a Beowulf clusters). Of these four architectures, in terms of cost, the two most popular parallel-processing paradigms are the SMP and cluster computers.



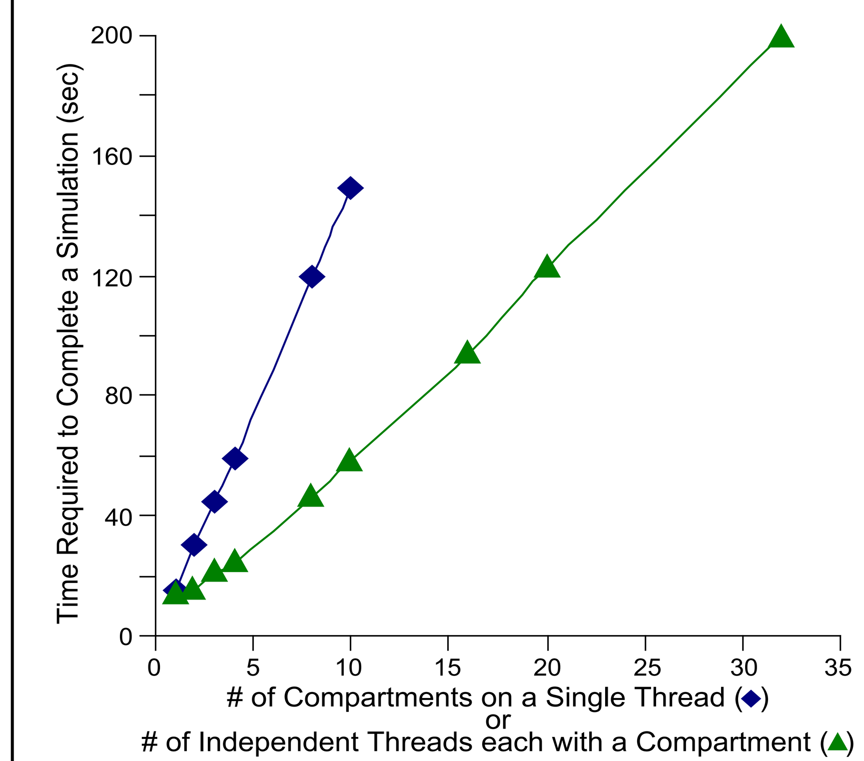
SMP machine architecture



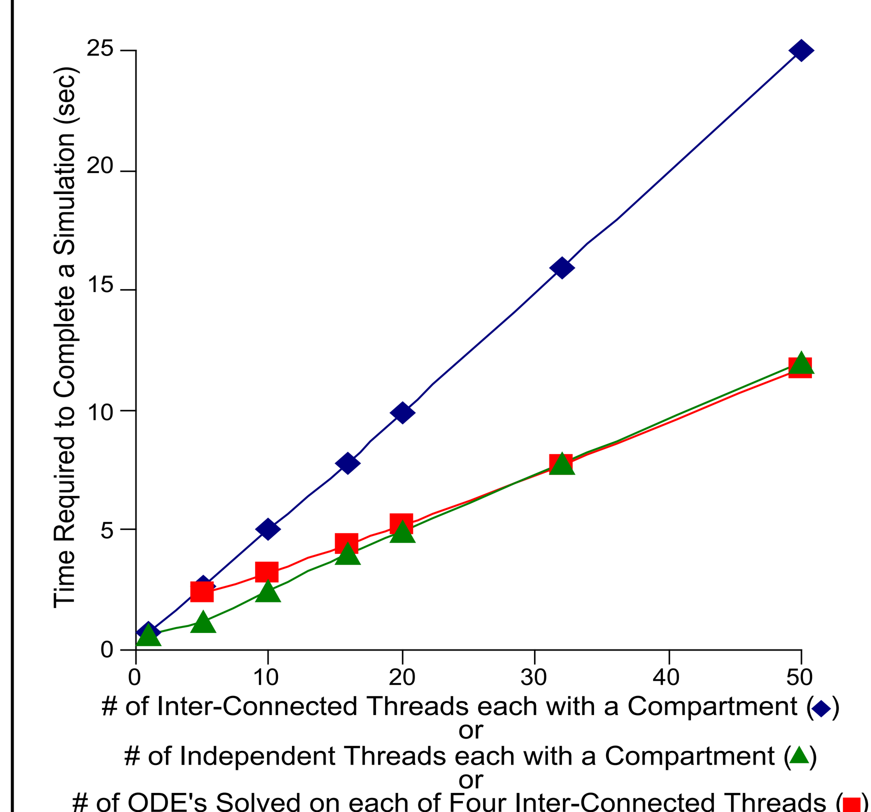
Beowulf cluster architecture

Simulation of load balancing and granularity

load balancing: Distributing processing activity evenly among several processors.
granulatiry: Dividing a processing task into smaller subtasks.



Gain from batch processing on a dual Xeon (hyperthread) computer. The number of compartments was systematically increased. With sequential processing (i.e., all compartments simulated on a single thread; blue) the time required to complete a simulation increased linearly with a slope of 15 sec/compartment. Simulating each compartment on a separate thread (i.e., parallel processing) produced an ~2.5-fold increase in performance (green).

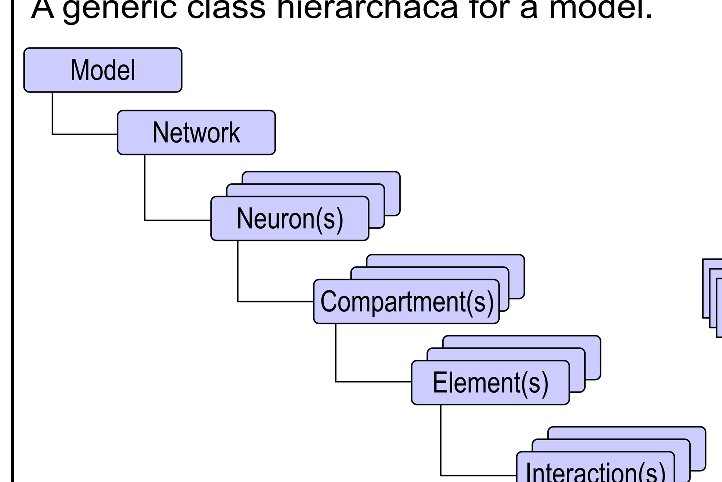


Comparing computation times for solving an equivalent number of ODEs, for multiple threads of coupled HH models (4 ODEs) with thread communication (blue), and multiple threads of uncoupled HH models (4 ODEs) with no communication among threads (green), and 4 threads with multiple (4-50) coupled ODEs (red). The green curve represents batch processing corresponding to the green curve in the upper figure. If the threads communicate (blue), then the benefits of parallel processing appear to be lost. However, with proper granularity (red) the benefits of parallel processing are realized, i.e., the ratio of computation time versus communication time increases.

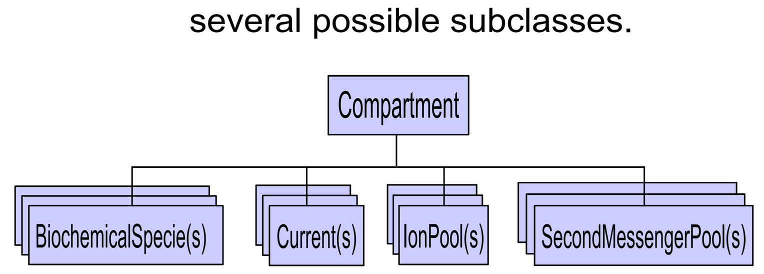
OO design of computaional model

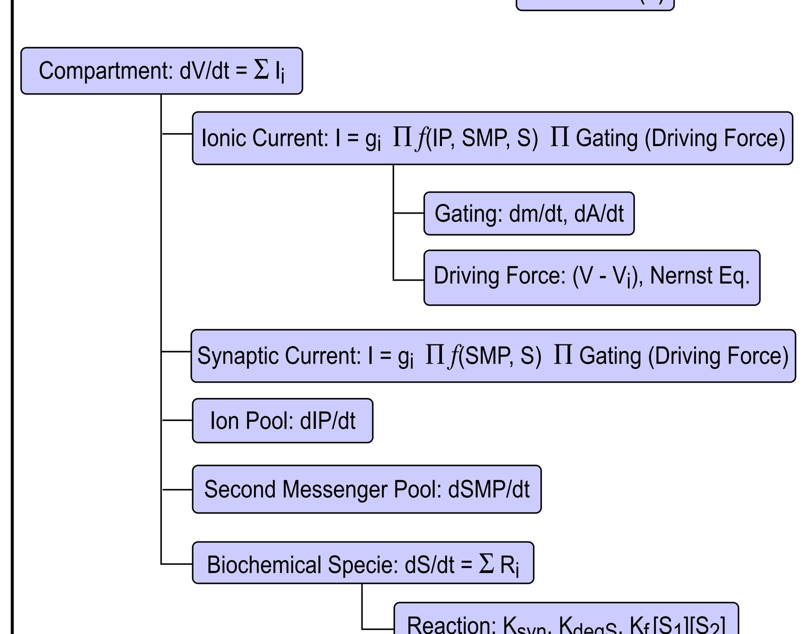
A object-oriented (OO) design uses encapsulation, inheritance and polymorphism which leads to modular components. Incorporating new components is therefore easier, and requires less debugging.

A generic class hierarchaca for a model.



A compatment class with several possible subclasses.





A compartment may contain numerous types of objects.

Hodgkin-Huxley model and compartmentalization

The Hodgkin-Huxley (HH) model (below, left) is represented by an object diagram (below,right). The compartment module contains the membrane potential ODE, while the activation and inactivation variables are defined in the base classes activation and inactivation.

$$C_m \frac{dV}{dt} = I - I_{Na} - I_K - I_l$$

$$I_{Na} = \bar{g}_{Na} m^3 h (V - V_{Na})$$

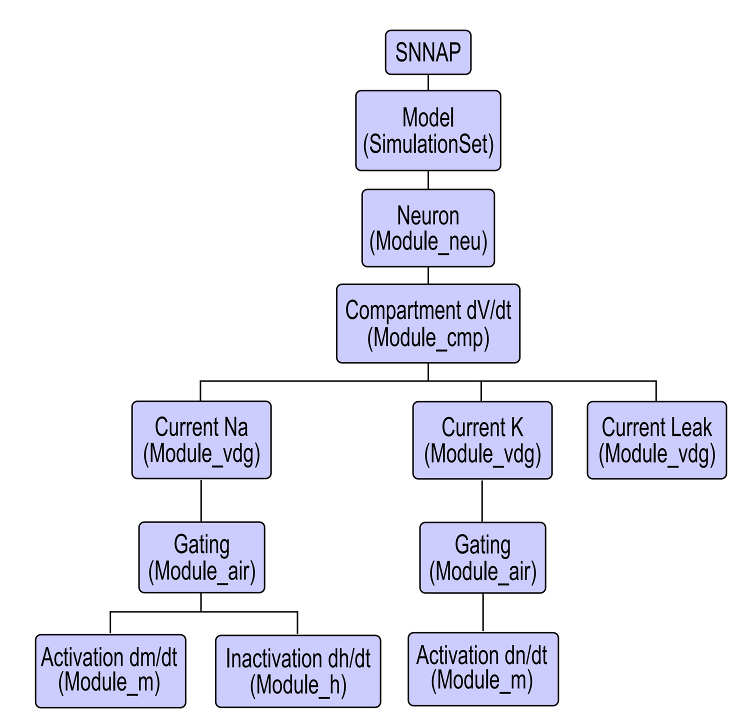
$$I_K = \bar{g}_K n^4 (V - V_K)$$

$$I_l = \bar{g}_L (V - V_L)$$

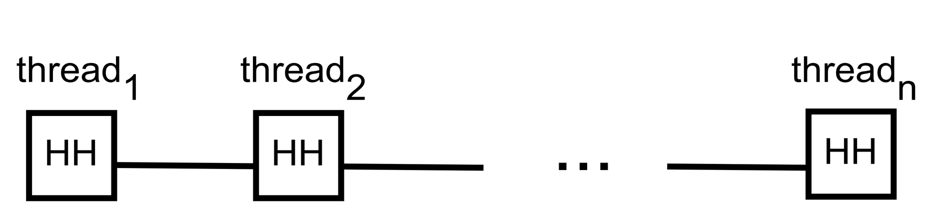
$$\frac{dm}{dt} = \phi[\alpha_m(V)(1-m) - \beta_m(V)m]$$

$$\frac{dh}{dt} = \phi[\alpha_h(V)(1-h) - \beta_h(V)h]$$

$$\frac{dn}{dt} = \phi[\alpha_n(V)(1-n) - \beta_n(V)n]$$



Coupling several HH models in series provides a test case for parallel processing.



Summary

We developed a prototype model to examine the costs/benefits of parallel processing. The prototype executed a variable number of coupled Hodgkin-Huxley models in parallel. The computational cost of parallel processing was due mainly to communication between threads. Tests identified two issues that determined the benefits from parallel processing: load balancing and model granularity. Load balancing relates to how many threads run efficiently on a single processor. Model granularity relates to how many ODEs are solved per thread. Compared to a nonparallel program (blue curve, upper figure above), a multithreaded program which ran on a dual Xeon processor computer had execution times reduced by 50% for up to 8 threads with 25 ODEs per thread (red curve, lower figure above). The intersection of the red and green curves in the figure above corresponds to the case when the computing time of 4 threads executing 25 ODEs (with communication) was equal to 25 threads executing 4 ODEs (without communication), i.e., 100 ODEs computed in total. The difference between the blue and green curves in the upper figure displays the gain due to parallel processing. With communication, the overall gain (difference between blue and red curves above) increased for larger numbers of ODEs, due to the increase in the ratio between computing time and communication time.

A parallel version of SNNAP will provide biologists with a useful tool for simulating complex models of neuronal, biochemical and molecular networks. With the decline in cost of multiple processor computers, the necessary hardware will be readily available.

Supported by NIH grants R01 RR11626 and P01 NS38310.