8.07f/W(Jul162004) +model NEUCOM : 10620 Prod.Type:FTP pp.1–5(col.fig.:NIL) ED:SreejaGA PAGN:bhaskara SCAN:

Available online at www.sciencedirect.com
ScienceDirect
Neurocomputing ▮ (▮▮▮▮) ▮▮▮–▮▮▮
NEUROCOMPUTING
www.elsevier.com/locate/neucom

Neurospaces:Towards automated model partitioning for parallel computers

Hugo Cornelis[a,*], Erik De Schutter[b]

[a]Research Imaging Centre, University of Texas Health Science Center at San Antonio, USA
[b]Laboratory of Theoretical Neurobiology, University of Antwerp, Belgium

Abstract

Parallel computers have the computing power needed to simulate biologically accurate neuronal network models. Partitioning is the process of cutting a model in pieces and assigning each piece to a CPU. Automatic partitioning algorithms for large models are difficult to design for two fundamental reasons. First, the algorithms must track the intrinsic asymmetries in the models and the dynamical behavior of the simulation. Second, the procedural nature of current modeling languages makes it difficult to extract the information needed by the algorithms.

From the start, the Neurospaces modeling system has been designed to deal with large and complicated neuronal models. The declarative nature of the software system allows to extract any kind of information from the model. In this work, we first show how to extract the information needed to partition a large model for simulation on parallel computers. Next, we use this information to compute a possible partitioning for a small and a large network model.
© 2006 Published by Elsevier B.V.

Keywords: Simulation; Middleware; Parallelization; Modeling

1. Introduction

It is well known that the simulation of biologically accurate neuronal models requires an enormous amount of computing power. With the recent technological advancements, nowadays CPUs have sufficient performance to meet the requirements of some single cell neuronal modeling and simulation. Nevertheless these single cell models must be put in the context of large scale neuronal networks to get realistic biological behavior [18]. As a result, to get adequate performance, the model must be run on parallel computers. Since some neurons are more complicated than others, biologically realistic neuronal network models have an intrinsic nature of heterogeneity. Hence, parallelizing a large and complicated neuronal network model becomes a difficult programming process and the need for automatic partitioning algorithms arises. Until today, no one has ever published a general automatic partitioning algorithm for biological accurate neuronal models. This work embodies the first necessary steps for such an algorithm.

2. The problem of defining models

Building complex and large neuronal models is a tedious process. Many biological components and physical parameters are involved and the relationships between them must be defined by the modeler. The number of, as well as the relationships between the biological components are the guiding entities for partitioning a large model over a parallel computer [22].

Neuron and Genesis share the idea that models covering various levels of complexity, are best described with a dedicated language for each level [2,15]. When using the Genesis neural simulator, an object-oriented scripting language defines models at the subcellular and network level, while a declarative language defines a cell's morphology. The Neuron simulator uses Hoc and Nmodl to define models. Hoc is an object-oriented scripting language for definition of network models. Nmodl has a semi-procedur-

*Corresponding author. Tel.: +1 210 5678112; fax: +1 210 5678152.
E-mail address: hygo.cornelis@gmail.com (H. Cornelis).

0925-2312/$ - see front matter © 2006 Published by Elsevier B.V.
doi:10.1016/j.neucom.2006.10.140

al nature and is used for intracellular mechanisms. As a result, there is no universal language for the descriptions of neuroscience models, that covers from the subcellular level up to the network level. Moreover, the languages are procedural or object-oriented in nature. This makes it very hard to extract any kind of data from a model description [14]. Automatic partitioning for parallel simulations of large models described in these languages, is very difficult, if not impossible.

3. Essentials of the neurospaces data model

Neurospaces is a modeling environment, designed with a focus on large and complicated models [7]. By leaving the simulation to a simulator, Neurospaces can be optimized for efficient memory handling, interaction with modelers and interaction with user interfaces [8]. As an identification service for neuronal simulators, Neurospaces is currently a unique middle-ware software component, comparable to a naming service for large software projects (e.g. JNDI [17]).

Neurospaces uses an abstract linked tree as a data model. The most important feature of this data model is that it is a natural abstraction of a neuroscience network model, because it easily accommodates tree shaped data with links between nodes at the same height of the tree. The nodes of the tree represent biological components, while the links between the nodes represent axonal connections. Neurospaces implements the abstract linked tree in a memory efficient way, and allows to restructure the model tree internally in anticipation of certain queries [5].

The used methodology allows automatic extraction of any kind of model information: firstly, all parameter specifications are enforced to be declarative. Secondly, parameterized algorithms are used instead of procedures. The parameterized algorithms accommodate both the layout of populations (see Fig. 2), as the connectivity of networks. Thirdly, when using a traditional object-oriented metaphor, it is often impossible to extract certain information from the classes. Therefore, Neurospaces does not make a distinction between objects and classes, but instead implements a template-instantiation pattern to allow parameter extraction from the templates as well as from the instances. This last process has the advantage that many parameters are stored only once, with possibly an encoding of a small deviation per instantiation.

The complexity of the core of Neurospaces and the internal data structures are hidden by traversal interfaces that allow to inspect a stored model in various ways. During the traversal, application callbacks see a sequence of small consistent model pieces and extract the required information for subsequent use.

4. Partitioning for parallel simulations

The implementation of the partitioning algorithm in Neurospaces consists of a single application callback that inspects the data stream generated using a traversal. The callback performs several operations on the data:

- The types of the modeling components are checked and candidate partitioning elements are tagged (the candidate partitioning types are currently fibers and cells, yet this is configurable). The candidate partitioning elements cluster the individual model pieces into candidate partitions.
- Each individual modeling piece is assigned a fixed actual workload. This allows the callback to compute the actual cumulative workload for each candidate partition.
- The expected workload per partition, defined as the total cumulative workload divided by the number of partitions, is compared with the actual cumulative workload of a candidate partition. As soon as the actual cumulative workload for a candidate partition gets greater than the expected workload per partition, the partition is sliced off from the model and defines a physical partition. The physical partition is assigned to a node of the parallel computer.

This algorithm defines a unique partitioning of a model.

5. An example

The granular layer of the cerebellar cortex contains granule cells and Golgi cells [21]. The first example we discuss contains 25 granule cells and two Golgi cells in a 2D grid layout. An outline picture using an abstract linked tree is shown in Fig. 1.

The granule cell and Golgi cell models are taken from [19]. Both models consist of a single compartment with a number of Hodgkin–Huxley alike channels and an exponentially decaying calcium pool. The granule cell population specification is shown in Fig. 2. By putting the cells in a 5 X 5 grid, 25 model cells are created. The Golgi cell population is specified in an analogue way.

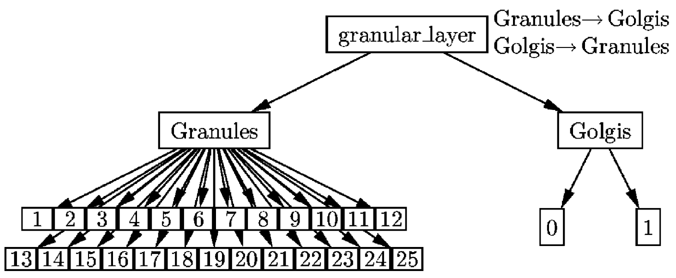


Fig. 1. Structure of the example network: at the right a Golgi cell population with two cells. At the left a granule cell population with 25 cells, in an evenly spaced rectangular grid of 5 × 5 with as center the midpoint of the two Golgi cells. The forward and backward projections are present at the level of the granular layer. The forward projection contains a set of connections between the granule cells and the Golgi cells. The backward projection goes from the Golgi cells to the granule cells.

```
PUBLIC_MODELS
  POPULATION GranulePopulation
    ALGORITHM Grid3D
      ALGORITHM_INSTANCE GranuleGrid
//          proto     #x     dx     #y      dy     #z   dz
      { Granule_cell  5     5e-05   5    3.75e-05  1    0.9}
    END ALGORITHM
  END POPULATION
END PUBLIC_MODELS
```

Fig. 2. Granule cell population specification. An algorithm is looked up from a library using a named reference. Next the algorithm is instantiated by providing the necessary parameters to run it. In this case the algorithm puts a template granule cell in a 5 X 5 grid in 3D space.

6. Workload computation

Hsolve is a compartmental solver that uses byte codes to encode the equations in a compartmental model [6]. Based on the experience obtained with these byte codes, each compartment in the model stored by Neurospaces can be assigned a load of 4 points, each channel a load of 6 points and a calcium pool a load of 3 points. A spike generating element as well as a synapse have a workload of 1 point. The total workload for a single granule cell model comes to 72 and for a single Golgi cell model to 51. The example network has a load of $1675 + 110 = 1785$.

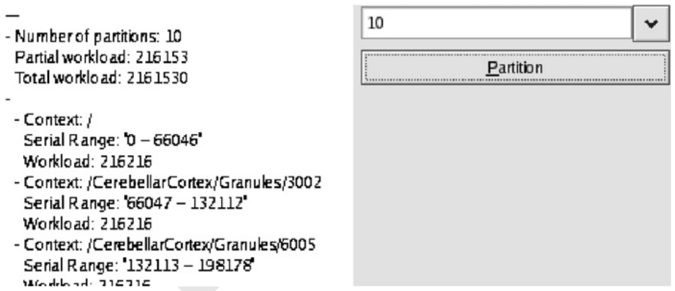
In [19] the ratio between granule cells and Golgi cells is about 1000. In Neurospaces, scaling up this small model to a more realistic size, is a matter of changing a couple of parameters. Scaling up this 2D network to 30 000 Granule cells and 30 Golgi cells [19] is accomplished by changing the population parameters in Fig. 2. If the network connectivity is specified using the right parameterization of connection algorithms, it automatically follows this scaling process. The memory efficiency of the data structures used in Neurospaces allow to store the large network model in little memory.

For the large network, the load for the granule cell population is 2160000 points while the load for the Golgi cell population is 1530 points, totaling 2161530 points. To partition this network model over many CPUs, the load has to be divided by the number of CPUs and rounded to a multiple of Golgi and granule cells. For 10 CPUs, we get an expected workload per CPU of 216153 points which is easily covered by assigning 3003 granule cells to the first 9 CPUs and have the last CPU compute 2973 granule cells and all the Golgi cells.

For 13 CPUs, we get a workload of 166271.5 points per CPU. The first 12 CPUs are assigned 2310 granule cells, while the last one is assigned 2280 granule cells and all the Golgi cells.

7. Compiling the model to a parallel simulation

Embedding Neurospaces in parallel Genesis [13] will allow to partition large models with Neurospaces and simulate them with parallel Genesis. Translating a large



```
—
- Number of partitions: 10
  Partial workload: 216153
  Total workload: 2161530
-
  - Context: /
    Serial Range: '0 – 66046'
    Workload: 216216
  - Context: /CerebellarCortex/Granules/3002
    Serial Range: '66047 – 132112'
    Workload: 216216
  - Context: /CerebellarCortex/Granules/6005
    Serial Range: '132113 – 198178'
```

Fig. 3. GUI for partitioning: a network of 30 000 granule cells and 30 Golgi cells over 10 partitions.

model described with Neurospaces into a parallel simulation involves three separate steps:

1. Apply the partitioning algorithm to compute the partitions and assign the partitions to the nodes of the parallel computer. In Neurospaces, the partitioning algorithm can be activated from a low-level command line interface or a high-level GUI (Fig. 3).
2. Initialize solvers on each node according to the assigned pieces of the model. This involves compilation of the model from the biological domain into the mathematical domain. This is explained in more detail in [6]. In Genesis/Neurospaces, this is done by invoking the Genesis action SETUP on the Neurospaces element (Fig. 4, a full explanation of the compilation is beyond the scope of this paper, for more information see [6] and [4]).

8. Related work

Partitioning of neuronal models is becoming increasingly popular. The following projects are closely related to this work or to Neurospaces as a software component:

- Neosim2 is a parallel discrete event simulator [16] and is based on the first Neosim project [12]. Neosim2 does not address the problem of heterogeneous models, but implicitly assumes homogeneous cell models. It therefore does not incorporate sophisticated partitioning

```
genesis > call models NEUROSPACES_SETUP \
                population /CerebellarCortex/Granules
genesis > reset
    time = 0.000000 ; step = 0
genesis > le / -t
    models {neurospacesmodel}         Granules {hsolvearray}
```

Fig. 4. Using Neurospaces with Genesis to simulate a part of the granular layer network. To instantiate the solvers, the SETUP action is invoked for every individual solver instance. In the last line, the Neurospaces element and the created solver instance are listed. All the other details like setting up connections according to the projections in the model, will be handled automatically.

algorithms as explained here.

- Other simulators like Topographica [1] and NEST [10] deal with more abstract neuron types like integrate-and-fire neurons. Due to the lower degree of heterogeneity for such systems, the problem of partitioning a large model has different characteristics.
- Catacomb is a simulator that deals with a range of simulation principles, and covers many biological levels. A notable feature of this simulator is that it, like Neurospaces, comes with a documented and well-thought out data model [3]. To the authors knowledge, the Catacomb simulator does not allow parallel simulations.
- The Neuron simulator has recently been expanded with a parallelization algorithm. The algorithm deals with conductance based neurons but does not address the problem of heterogeneity, yet the parallelization of the simulation is completely transparent [20].
- Neuroconstruct is a modeling environment that interfaces with the Neuron and Genesis simulators [11]. After creation of a network model, scripts for the appropriate simulator are generated. Neuroconstruct has been designed to allow the construction of networks and to make them more accessible. Therefore, it comes with a rich GUI and a nice rendering engine. How well the approach scales, is currently an open question. The interface with the simulator is not interactive, which might be a problem for large simulations. Neurospaces targets large heterogeneous models. Scaling problems have been avoided by choosing carefully designed algorithms and datastructures, that allow to optimize the performance of Neurospaces [8]. Neurospaces has been embedded in Genesis for maximum interfacing capabilities with the core of the simulator.

Many things are moving in the software that is guiding theoretical computational neuroscience through its activities. Yet, from the above comparison, it is clear that parallelization of simulations remains a hard to understand problem.

9. Conclusion and future work

Neurospaces automatically assembles all descriptive parameters of neuronal models. This allows to compute the CPU load of a simulation and to partition a model such that it is ready for parallel simulations.

The single compartment granule and Golgi cells are only moderately asymmetric. Introducing computationally more intensive Purkinje cell models (e.g. [9]) in the network will result in huge asymmetries in the network model. A logical next step is to evaluate and improve the algorithm performance when dealing with such asymmetrical network models.

It is important to emphasize that this paper is only the first step towards extended partitioning algorithms. Further steps will include the following:

- The network connectivity is currently ignored. Although including the connectivity in the algorithm is a necessary step, it transforms the nature of the complexity of the problem from that of a searching problem (with linear scaling) to the complexity from a matching problem (with exponential scaling). Then, after including the connectivity in the problem, we are dealing with an optimization problem that cannot be solved deterministically, such that heuristics need to be applied. The result of the algorithm proposed in this paper provides a reliable data set for heuristic partitioning algorithms.
- The communication load via the connections between different partitions of the model must be included in any partitioning algorithm, thereby further complicating the problem.
- The proposed algorithm determines a fixed workload, and runs before the actual simulation. To enable dynamic load balancing and computing nodes to join and leave a simulation, the algorithm needs to monitor the current load of all CPUs and adapt the partitioning dynamically.

Each of these steps requires more research work.

References

[1] J.A. Bednar, Y. Choe, J. De Paula, R. Miikkulainen, J. Provost, T. Tversky, Modeling cortical maps with Topographica, Neurocomputing (2004) 1129–1135.
[2] J.M. Bower, D. Beeman, (Eds.), The Book of GENESIS, second ed. Springer, Berlin, 1998.
[3] R.C. Cannon, M.E. Hasselmo, R.A. Koene, From biophysics to behavior: catacomb2 and the design of biologically plausible models for spatial navigation, Neuroinformatics 1 1 (2003).

[4] H. Cornelis, The interference between modeling space and simulation space, fundamental techniques for efficiency in large and complicated simulations, Ph.D. Thesis, UIA, University of Antwerp, 2004.
[5] H. Cornelis, E. De Schutter, Treespaces, Technical report, UIA, University of Antwerp, Universiteitsplein 1, Wilrijk, Antwerp, December 2001.
[6] H. Cornelis, E. De Schutter, Tutorial: simulations with genesis using Hsolve, Course material, November 2002, available from ⟨http://www.genesis-sim.org⟩.
[7] H. Cornelis, E. De Schutter, Neurospaces: Separating modeling and simulation, Neurocomputing 52–54 (2003) 227–231.
[8] H. Cornelis, E. De Schutter, Neurospaces parameter handling, Neurocomputing 58–60 (2004) 1079–1084.
[9] E. De Schutter, J. Bower, An active membrane model of the cerebellar purkinje cell I. simulation of current clamps in slice, J. Neurophysiol. 71 (1994) 375–400.
[10] M. Diesmann, M.-O. Gewaltig, NEST: An environment for neural systems simulations, in: T. Plesser, V. Macho (Eds.), Forschung und wisschenschaftliches Rechnen, Beiträge zum Heinz-Billing-Preis 2001, GWDG-Bericht. Ges. für Wiss. Datenverarbeitung, vol. 58, Göttingen, 2002, pp.43–70.
[11] P. Gleeson, Building 3D network models with neuroConstruct, World Wide Web, March 2005, Tutorial at the Wam-Bam meeting, ⟨http://wam-bamm.org/WB05/Tutorials/advanced-tutorials/gleeson/index.html⟩.
[12] N. Goddard, G. Hood, F. Howell, M. Hines, E. De Schutter, Neosim: portable plug and play neuronal modelling, Neurocomputing 38–40, 1–4 (2001) 1657–1661.
[13] N.H. Goddard, G. Hood, Large-scale simulation using parallel GENESIS. The Book of GENESIS, second ed. Springer, Berlin, 1998, (Chapter 21).
[14] N.H. Goddard, M. Hucka, F. Howell, H. Cornelis, K. Shankar, D. Beeman, Towards NeuroML: Model description methods for collaborative modelling in neuroscience, Philos Trans. Roy. Soc. Ser. B: Biol. Sci. 356 (2001) 1–20, Theme Issue organized and edited by Rolf Kötter on "Neuroscience databases—tools for exploring brain structure-function relationships".
[15] M. Hines, N. Carnevale, The NEURON simulation environment, The Handbook of Brain Theory and Neural Networks, second ed. MIT Press, Cambridge, MA, 2003, pp. 769–773.
[16] F. Howell, R. Cannon, N. Goddard, H. Bringmann, P. Rogister, Neosim2, World Wide Web, September 2005, ⟨http://www.neurogems.org/neosim2/⟩.
[17] F. Howell, R. Cannon, N. Goddard, H. Bringmann, P. Rogister, H. Cornelis, Linking computational neuroscience simulation tools: a pragmatic approach to component-based development, Neurocomputing 52–54 (2003) 289–294.
[18] F. Howell, J. Dyhrfjeld-Johnsen, R. Maex, N. Goddard, E. De Schutter, A large-scale network model of the cerebellar cortex using pgenesis, Neurocomputing 32 (2000) 1041–1046.
[19] R. Maex, E. De Schutter, Synchronization of golgi and granule cell firing in a detailed network model of the cerebellar granule cell layer, J. Neurophysio. 80 (1998) 2521–2537.
[20] M. Migliore, C. Cannia, W. Lytton, H. Markram, M.L. Hines, Parallel network simulations with neuron, J. Comput. Neurosci. (2006). PMID: 16732488.
[21] S.L. Palay, C.-P V., The Cerebellar Cortex: Cytology and Organization, Springer, Berlin, 1974.
[22] M.C. Peter Pacheco, T. Uchino, Parallel neurosys: a system for the simulation of very large networks of biologically accurate neurons on parallel computers, Neurocomputing 32–33 (2000) 1095–1102.