Cogn Comput
DOI 10.1007/s12559-016-9411-y

Parallel Brain Simulator: A Multi-scale and Parallel Brain-Inspired Neural Network Modeling and Simulation Platform

Xin Liu[1] · Yi Zeng[1,2] · Tielin Zhang[1] · Bo Xu[1,2]

Received: 10 October 2015 / Accepted: 8 April 2016

Abstract The brain is naturally a parallel and distributed system. Reverse engineering a cognitive brain is considered to be a grand challenge. In this paper, we present the parallel brain simulator (PBS), a parallel and distributed platform for modeling the cognitive brain at multiple scales. Inspired by large-scale graph computation, PBS can be considered as a universal parallel execution engine, which is aimed at reducing the complexity of distributed programming and providing an easy to use programmable platform for computational neuroscientists and artificial intelligence researchers for modeling and simulation of large-scale neural networks. As illustrative examples and validations, three brain-inspired neural networks which are built on PBS are introduced, including the 1:1 human hippocampus network, the 1:1 mouse whole-brain network and the CASIA brain simulator built for cognitive robotics. We deploy PBS on both commodity clusters and supercomputers, and a scalable performance is achieved. In addition, we provide evaluations on the scalability and performance of both lumped synapse-based simulation and non-lumped synapse-based simulation with different data-graph distribution methods to show the effectiveness and usability of the PBS platform.

Keywords Neural network simulator · Graph computation · Parallel simulation · Spiking neural network · PBS

Introduction

The brain is a complex nonlinear system. Computational modeling and simulation of the brain helps to understand its information processing principles and provides inspirations for creating next-generation intelligent systems. Since "the large-scale models can help us to test large-scale hypotheses, which can address what makes us humans" [13], one of the trends for brain simulation is to build very large-scale, even whole-brain scale, neural networks at multiple levels of details. The human brain, one of the world's most complicated systems, consists of about 10^{11} neurons, each of which owns 10^3 to 10^4 synapses [3]. The simulation of neural networks on such a scale is very challenging due to the required memory to represent the network structures and the dynamics of the neural network systems and, consequently, can practically be done only by exploiting the computing power and the memory of multiple computation systems by means of a distribution. Recent advances in computer hardware and parallel computation have made the simulation of neural networks with millions or even billions of neurons possible.

Various efforts on creating neural networks simulation platforms have been made. The work introduced in [6] and the Web site[1] provide relatively comprehensive overviews.

Xin Liu and Yi Zeng contributed equally to this work and serve as co-first authors.

✉ Yi Zeng
yi.zeng@ia.ac.cn

1 Research Center for Brain-inspired Intelligence, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China

2 Center for Excellence in Brain Science and Intelligence Technology, Chinese Academy of Sciences, Shanghai 200031, China

[1] https://grey.colorado.edu/emergent/index.php/Comparison_of_Neural_Network_Simulators.

Although having been initially envisioned for a specific purpose or domain of applicability, many of them are extended to improve their generality and to support modeling various neural networks at different scales. These platforms can be generally classified into three types. The first type is built for simulating detailed multi-compartmental neuron models and small-scale networks, including NEURON [7] and GENESIS [4]. Another type of relatively more general platforms focuses on simulating large-scale neural networks and the improvement of their simulation efficiency through parallel computation, including NEST [14], NCS [6], SPLIT [17], PCSIM [29], C2 [2] and BRIAN [16]. In particular, by using NEST on K computers [33], Kunkel et al. [21] achieved a maximum network size of 1.86×10^9 neurons and 11.1×10^{12} synapses, which is the largest simulation of spiking neural network reported so far. The third type is more dedicated, such as iNVT[2] for brain-inspired neuromorphic vision, NENGO for modeling functional brain based on Neural Engineering Framework [12] and Emergent[3] for deep neural networks. In addition, all of the above platforms support either of the two synapse model types: the lumped synapse models [30] and the non-lumped (or standard) synapse models [27]. In short, the lumped synapse models "neglect all the spatial structure of a neuron and assume the system of equations governing a synapse as linear, and collapse the state variables of all synapses with identical dynamics into the state variables of a single lumped synapse" [23, 30], while the non-lumped synapse models are "derived from the biophysics, and the synaptic dynamics in these models are governed by differential equations and the state variables fully capture the complete history of all synaptic events" [23].

In this paper, we present the parallel brain simulator (PBS), a parallel and distributed simulation platform which is designed for modeling and simulating multi-scale cognitive brain (from ions, neurons, neural micro-circuits and meso-circuits, all the way to brain regions, macro-circuits and cognitive behaviors, and from several neurons to billions of neurons, while each of them is with approximately 10^3 to 10^4 synapses). From the architecture perspective, PBS is inspired by the large-scale computational platforms such as MapReduce [9] and graph computation platforms such as Pregel [25] and PowerGraph [15]. It can be considered as a universal execution engine for brain-inspired neural network simulations. The PBS platform aims at reducing the complexity of distributed programming for neuroscientists and artificial intelligence researchers, etc. PBS is developed on both commodity clusters and supercomputers. The experimental results show that the platform achieved scalable performance for simulating multi-scale neural networks. Furthermore, since PBS provides a programmable platform between the high-level simulation by using Python scripts and the low-level simulation by using distributed programming language, a wide range of neural networks can be modeled and simulated on this platform. To validate the platform, several neural networks are built and simulated on PBS, and three of them are introduced in this paper, including the human hippocampus network, the mouse whole-brain network and the CASIA brain simulator used on a series of cognitive robotics tasks[4] The major contributions of this paper are summarized as the following:

- A novel method for simulating brain-inspired neural networks at multiple scales. To the best of our knowledge, PBS is among the first to introduce graph computation into the area of brain simulation.
- A prototype platform as an implementation of the simulation method, exploiting its architecture on both commodity clusters and supercomputers. Up to now, we have implemented a lumped synapse-based network with 2.11×10^8 neurons and 5.91×10^{11} synapses, and a non-lumped synapse-based network with 1.01×10^7 neurons and 3.72×10^{10} synapses.
- Several useful features, which make simulations visible and controllable. They include the real-time neuronal/synaptic states recording and visualization, the real-time human–simulator interaction and the step-by-step (the step is a time interval of the simulation) execution of the simulation.
- Three validation cases. The 1:1 human hippocampus network and the 1:1 mouse whole-brain network are built and simulated, and the CASIA brain simulator, which is inspired by the cognitive behavior of the human brain, is built on PBS.
- Three novel distribution methods proposed for distributing the computational burden during neural network simulation.
- Evaluations of the scalability and the performance of both the lumped synapse-based simulations and the non-lumped synapse-based simulations with different distribution methods.

PBS: An Overview

The PBS platform is a distributed execution engine for modeling and simulating multi-scale neural networks. In this section, we will give a general introduction to this platform. Firstly, we present the neural networks abstraction (NNA), which is the core abstraction that PBS supports. Then we introduce the parallel abstraction, the

[2] http://ilab.usc.edu/toolkit/home.shtml.

[3] https://grey.colorado.edu/emergent/.

[4] http://bii.ia.ac.cn/brainbo.html.

concrete architecture of a PBS cluster for distributed execution and the implementation of the platform. Finally, we introduce other concepts on how PBS executes a simulation represented as an NNA.

Neural Network Abstraction

As defined by Aleksander and Morton [1], neural computing can be considered as "the study of networks of adaptable nodes which, through a process of learning from task examples, store experiential knowledge and make it available for use." Hence, the generalization and abstraction of neural networks are similar to the graph under the view of architecture or topology. Following (Aleksander and Morton [1]), we define the neural network abstraction (NNA), the core abstraction of PBS at a certain scale, as a six tuple:

$$\mathrm{NNA}^{(s)} = \langle V^{(s)}, E^{(s)}, \mathrm{IF}^{(s)}, \mathrm{OF}^{(s)}, \mathrm{WA}^{(s)}, \mathrm{OA}^{(s)} \rangle$$

where $V^{(s)} = \{v_i | i = 1, 2, \ldots, N\}$ refers to the vertex group, and $E^{(s)} = \{e_j | j = 1, 2, \ldots, M\}$ refers to the edge group, each element of which represents a connection between two vertexes. Each vertex or edge in these groups can represent different objects. Between any two vertexes, multiple edges can be defined in $E^{(s)}$, which is an important property for synapse modeling. For instance, to model a network at a finer level of details, $V^{(s)}$ represent the group of complex compartmental Hodgkin–Huxley neurons [18], while $E^{(s)}$ denotes the group of conductance-based synapses with ligand as well as voltage-gated properties and with different forms of plasticity. Even intracellular processes in the form of gene regulatory and biochemical signaling networks can be incorporated into a detailed neuron model represented by v_i. Besides, since v_i can be used to represent various objects at different scales, the network may comprise several different neuron types with the typical blend of ion channels. At a coarser level of details, to model a network with neurons of single-compartment reduced model and synapses of simple current of conductance is also available on PBS by properly defining $V^{(s)}$ and $E^{(s)}$. At an even coarser level, the networks may be built based on a unit, intended to represent, for instance, a local population of several hundred neurons. Going a level up further, an entire cortical column or cortical region may be represented by a single unit. These coarse-grained networks can also be built on PBS by denoting $V^{(s)}$ as a group of corresponding units. $\mathrm{WA}^{(s)} : \mathrm{IF}^{(s)} \rightarrow \mathrm{OF}^{(s)}$ refers to the working algorithm, denoting the information processing procedure of neural networks, $\mathrm{OA}^{(s)} : \langle V^{(s)}, E^{(s)} \rangle \rightarrow \langle V^{(s)}, E^{(s)} \rangle$ refers to the self-organization algorithms, denoting the learning procedure. $\mathrm{IF}^{(s)} \subseteq V^{(s)}$ and $\mathrm{OF}^{(s)} \subseteq V^{(s)}$ refer to the input field and the output field of the NNA, respectively. As an example, the NNA of mouse brain network introduced in "Mouse Whole-Brain Simulation" section is shown in Fig. 2.

Parallelization Abstraction

In order to make the NNA programmable and executable, an extended directed acyclic graph (DAG) abstraction [24] is used as the parallelization abstraction of PBS. This abstraction consists of a sparse data graph $G = \langle V, E \rangle$, and two independent vertex-programs Q_w and Q_a, which can be executed in parallel on each vertex $v \in V$ and can interact with the neighboring instances $Q_\mathrm{w}(v)$ or $Q_\mathrm{a}(v)$, where $(u, v) \in E$. The data graph G encodes both the NNA's specific sparse computational structure and modifiable program states. The users can associate arbitrary blocks of data (or parameters) to the vertexes and the edges of G. Specifically, a vertex can be used to save a hybrid NNA or a neuron, and an edge can be used to save a connectome or a synapse. In addition, the parallelization abstraction extends the DAG abstraction by employing two vertex-programs, used for WA and OA, respectively. The interaction between vertex-program and graph structure enables the optimization of data layout and communication. The abstraction also provides interface to configure the IF field and the OF field.

The abstraction of vertex-programs, the Q_w and the Q_a, uses the GAS model [15], where the vertex-program is divided into three conceptual phases, namely Gather, Apply and Scatter. In the Gather phase, the information about the adjacent vertexes and edges is collected through a generalized sum over the neighborhood of the vertex u on which the $Q_\mathrm{w}(u)$ and the $Q_\mathrm{a}(u)$ are running. The resulting value *acc* is used in the Apply phase to update the value of the central vertex. Finally, the Scatter phase uses the new value of the central vertex to update the data on the adjacent edges. The fan-in and the fan-out of a vertex-program are determined by the corresponding Gather and Scatter phases. As an example of applying the vertex-program, the Q_w and the Q_a, used in the mouse brain network presented in "Mouse Whole-Brain Simulation" section, are shown in Fig. 3b, c.

System Architecture

The PBS cluster consists of three types of components, including the nna-master, the nna-servers and the vtx-peers, as shown in Fig. 1. The nna-master is a single master, which coordinates the nna-servers to execute individual simulations. Since the parallelization abstraction relies on the distributed data graph to store the computational state and encode the interaction between vertex-programs, the nna-master is also used to distribute the data-graph structure G on

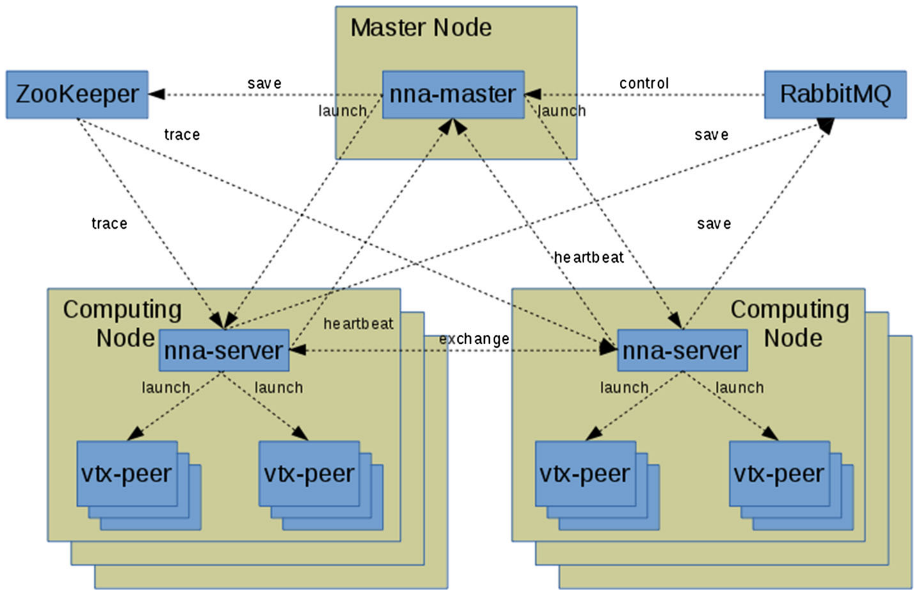


Fig. 1 The system architecture of PBS

the nna-servers. The nna-servers manage both the data-graph structure and the vertex-peers and drive the vertex-peers to execute the tasks of the vertex-programs. In the beginning, the nna-servers register themselves to the nna-master and periodically send a heartbeat to demonstrate their continued availability. After the data graph is distributed to the nna-servers by the nna-master, the appropriate vertex-peers are invoked. A vertex-peer is a generic component that prepares the input data and invokes the computation on it, typically by executing an external process. Besides these components, there are two distribution services used in PBS, including ZooKeeper[5] and RabbitMQ.[6] ZooKeeper is used to track the states of nna-servers and store the replicas of the data graph. RabbitMQ is used for real-time input/output, states storage and analysis and debugging. In addition, there are one or more clients (not shown in the figure). The client submits a neural network to the nna-master and either polls the master to discover the job status or blocks until the job completes.

Implementation

Considering simulation efficiency, PBS applies an in-memory computing method. The architecture is implemented by using a three-level parallel programming model: (1) SIMD processing in the core; (2) thread programming in a computing node using Pthreads; (3) distributed-memory parallel programming with MPI. All the data used during simulation are stored on RAM, where the bulk data (data graph) are stored on the computing nodes, and the master handles references. The local hard disks are used only to store the snapshot of the data graph (see "Fault Tolerance, Snapshot and Others" section for details). In addition, since the distribution of data graph on computing nodes reduces the computing burden on single node, the distribution method plays a key role on the scalability and the performance for the simulation of large-scale networks. We will discuss the distribution method in "Simulations on PBS" section (at the end of "Mouse Whole-Brain Simulation" and "Human Hippocampus Network Modeling" sections) and "Scalability and Performance" section (focusing on scalability and performance).

Synchronization

In order to simulate the neural network activities in different time intervals, the platform applied a synchronous execution engine, where all the active vertex-programs are executed synchronously in a sequence of super steps (iterations) in both the shared and distributed-memory settings. Each iteration simulates one simulation tick (minimum time interval of the simulation), which is set to be 1 ms by default. Synchronization is achieved by a barrier at the end of each iteration. The simulation of one iteration can be executed either synchronously or asynchronously, leading to the different trade-offs in simulation performance, system performance and determinism. For the synchronous method, the platform executes the phases of vertex-programs, which are the Gather phase, the Apply phase and the Scatter phase, respectively, and each phase runs synchronously on all active vertexes with a barrier at the end. In contrast, for the asynchronous method, the platform executes active vertexes as a processor and the network resources become available immediately. Namely, the simulation needs to firstly obtain the mutex lock before updating the vertexes and the edges during the Apply and the Scatter phases, and then, it commit

5 http://hadoop.apache.org/zookeeper.

6 http://www.rabbitmq.com.

the update to the graph and make the update visible to the subsequent computations on the neighboring vertexes.

There are two types of algorithms for simulation of neural networks: the clock-driven algorithms, with which all the neurons and synapses are updated simultaneously at every tick of a clock, and the event-driven algorithms, with which the updated neurons and the updated synapses are only those which receive or emit a spike (hybrid strategies also exist) [6]. For simulation of very large-scale networks, the simulation time with both of these algorithms scales as the total number of spike transmissions, but each algorithm has its own advantages and disadvantages. The PBS platform provides a convenient interface for users to implement both of these algorithms, namely, the message sending type at the Scatter phase of the vertex-program. The simulations with the clock-driven algorithm need to make sure all the vertexes accept at least one message at every tick, while during the simulation with the event-driven algorithm, only the vertexes with an active neighborhood accept the messages.

Fault Tolerance, Snapshot and Others

Similar to the graph computation platforms such as Pregel and PowerGraph, PBS achieves fault tolerance by saving a snapshot of the data graph to the hard disks. The execution engine constructs the snapshot at every time interval (which is configurable). The checkpoint overhead, typically a few seconds for the very large-scale simulations we did, takes a very small fraction of the whole simulation time. The snapshot method is also configurable to record the states of the specific vertexes and/or edges. In addition, PBS uses the message queue service, RabbitMQ, to implement the real-time interaction between the clients and the simulations, such as the visualization, the remote states recording and the execution control. Particularly, the interaction is used to simulate the external stimulus to the simulation in real time and used to implement the step-by-step execution for debugging. Furthermore, lots of distributed execution tools for analysis of the simulations are provided by the platform.

Simulations on PBS

Several simulations of biological or brain-inspired neural networks were modeled and simulated on PBS, and three of them will be introduced in this section, including the 1:1 mouse whole-brain network, the 1:1 human hippocampus network and the CASIA brain simulator. The first two models were built based on the brain architectures from anatomical experiments and the state-of-art EGFP (enhanced green fluorescent protein) experiments [28]. The mouse brain network shows PBS's ability to simulate whole-brain scale neural networks, and the hippocampus network shows the ability to simulate fine-grained brain networks with more neuronal and synaptic details. The CASIA brain was built on PBS, and it validates PBS's ability for creating functioning brain-inspired neural networks.

All the simulations were performed on two distributed-memory clusters at the super computing center affiliated to Institute of Automation, Chinese Academy of Sciences, Beijing, China. One cluster named the blade cluster is composed of 16 nodes (blade server), and each of it has 6 Intel(R) Xeon(R) CPU E5-2620 processors with 24 cores and 64 GB of memory. The other cluster named the fat cluster is composed of 16 blade nodes and 2 "fat" computing nodes, and each of the "fat" node has 4 Intel(R) Xeon(R) CPU E7-4820 processors with 64 cores and 1 TB of total memory. The computing nodes of both clusters are connected with gigabit ethernet, the bandwidth of which is 1 GB/s.

Mouse Whole-Brain Simulation

According to the state-of-art EGFP experiment, 213 brain regions of the mouse brain and the meso-scale connectome among these regions were reported [28]. By applying the multi-scale connectome transformation method on the EGFP results [34], we obtained the approximate amount of cells per region and the approximate number of synaptic connections between two connected regions (namely, the micro-scale connectome which is represented as $M_{ij}(1 \le i,j \le 213)$). In order to simulate the whole-brain scale network based on $M_{ij}(1 \le i,j \le 213)$, we firstly built the neural network abstraction. The NNA (shown in Fig. 2) consists of three layers: the brain layer, the region layer and the neuron layer. The top is the brain layer with one NNA$^{(\text{brain})}$:

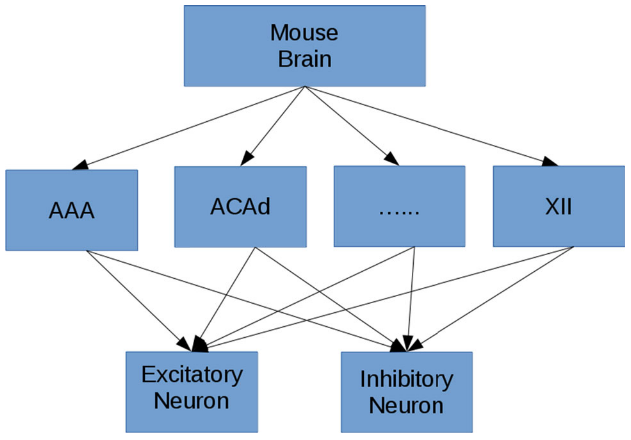


Fig. 2 The NNA for the mouse brain network

$$\mathrm{NNA}^{(\mathrm{brain})} = \langle V^{(\mathrm{brain})}, E^{(\mathrm{brain})}, \mathrm{IF}^{(\mathrm{brain})}, \mathrm{OF}^{(\mathrm{brain})}, \mathrm{WA}^{(\mathrm{brain})}, \mathrm{OA}^{(\mathrm{brain})} \rangle$$

$$V^{(\mathrm{brain})} = \left\{ \mathrm{NNA}^{(\mathrm{AAA})}, \mathrm{NNA}^{(\mathrm{ACAd})}, \ldots, \mathrm{NNA}^{(\mathrm{XII})} \right\},$$

where each vertex is itself an NNA of one of the 213 brain regions. $E^{(\mathrm{brain})} = \{e_{ij}^{(\mathrm{brain})} | 1 \le i, j \le 213, M_{ij} > 0\}$. Namely, any two regions with synaptic connections have one e_{ij} in $E^{(\mathrm{brain})}$. Both $\mathrm{WA}^{(\mathrm{brain})}$ and $\mathrm{OA}^{(\mathrm{brain})}$ have two functions, namely, the $\mathrm{WA}^{(r)}$ (r represents the regions) driver and the states recorder. $\mathrm{IF}^{(\mathrm{brain})}$ represents the inputs of the network, including an $\mathrm{NNA}^{(\mathrm{VISp})}$ (visual input region VISp), an $\mathrm{NNA}^{(\mathrm{AUDp})}$ (auditory input region AUDp) and several $NNA^{(\mathrm{SSp}-*)}$ (somatosensory input regions, including SSp-bfd, SSp-ll, SSp-m, SSp-n, SSp-tr, SSp-ul and SSp-un.). $\mathrm{OF}^{(\mathrm{brain})}$ includes $\mathrm{NNA}^{(\mathrm{MOp})}$ and $\mathrm{NNA}^{(\mathrm{MOs})}$ (motor output).

Under the whole-brain layer is the brain region layer, and its abstraction is represented as:

$$\mathrm{NNA}^{(r)} = \langle V^{(r)}, E^{(r)}, \mathrm{IF}^{(r)}, \mathrm{OF}^{(r)}, \mathrm{WA}^{(r)}, \mathrm{OA}^{(r)} \rangle,$$

$$V^{(r)} = \left\{ n_i^{(\mathrm{ex})}, n_j^{(\mathrm{in})} | 1 \le i \le N^{(\mathrm{ex})}, 1 \le j \le N^{(\mathrm{in})} \right\},$$

where r represents different brain regions (e.g., AAA, ACAd, ACAv,...), $n_i^{(\mathrm{ex})}$ and $n_j^{(\mathrm{in})}$ represent the vertexes which are excitatory neurons and inhibitory neurons, respectively. The lumped synapse model and the non-lumped synapse model are both available for $E^{(r)}$. Major consideration for the choice of these models is the hardware resources. For the lumped synapse model-based NNA,

$$E^{(\mathrm{lumped}-r)} = \left\{ e_{ij}^{(\mathrm{lumped}-r)} | 1 \le i, j \le N^{(r)} \right\},$$

where $e_{ij}^{(\mathrm{lumped}-r)}$ is an edge with a lumped synapse between n_i and n_j. For the standard synapse-based NNA,

$$E^{(\mathrm{standard}-r)} = \left\{ e_{ijk}^{(\mathrm{standard}-r)} \middle| 1 \le i, j \le N^{(r)}, 1 \le k \le N_{ij}^{(r)} \right\},$$

where $e_{ijk}^{(\mathrm{standard}-r)}$ is an edge with a synapse between n_i and n_j (n_j can be a neuron in other regions). $\mathrm{WA}^{(r)}$ is an abstraction of information processing procedure, and $\mathrm{OA}^{(r)}$ denotes an STDP-based algorithm [32]. In addition, both $\mathrm{IF}^{(r)}$ and $\mathrm{OF}^{(r)}$ are set randomly.

After the NNA setup, we simulate the model on the PBS platform, where each part of the $\mathrm{NNA}^{(\mathrm{brain})}$ is coded as a C++ class. Most of these classes are simply and directly coded and can be set as default instead of writing many additional codes for them. The special ones are Q_{w} for $\mathrm{WA}^{(r)}$ and Q_{o} for $\mathrm{OA}^{(r)}$, where the GAS model is used. As an illustrative example, some codes of the classes are listed in Fig. 3.

Within the simulations on PBS, the distribution method can be easily implemented by using the function add_vertex (*idx*, placement, ...) and add_edge (*idx*$_1$, *idx*$_2$, placement, ...), where *idx*, *idx*$_1$, and *idx*$_2$ are the indexes of neurons, and the *placement* is the index of computing nodes. For the simulation of mouse whole-brain network, we exploit three distribution methods, including the random method, the region-based method and the cluster-based method. We also have evaluated the effectiveness for each of the proposed methods, which will be introduced in "Scalability and Performance" section.

As the name implies, with the random distribution method, the simulation randomly distributes neurons and synapses on computing nodes, ignoring the information of regions and connectome. Consequently, the way on the number of neurons distributed per node is calculated as:

$$\mathrm{MNE}_{\mathrm{avg}} = \frac{\sum \mathrm{NE}_i}{M}, \tag{1}$$

(a)
```
// base neuron class
class neuron:
  states[]; init(); reset(dt);

  //! update the neuron states
  advance(t)
  //! set the input current
  input(inj)
  //! get the membrane voltage
  output()
  //! is the neuron spiked?
  spikeHit(t)

// base synapse class
class synapse:
  ...
  //! update the synapse
  update()
  //! synapse transmission delay
  delay()
```

(b)
```
// gather_nbrs: NONE_NBRS
gather(D_u, D_uv, D_v):
  return NULL

apply(D_u, acc):
  return NULL

// scatter_nbrs: IN_NBRS
scatter(D_u, D_uv, D_v):
  // update the synapse
  // if Du is the target
  if(D_u.isTgt()):
    D_uv.update()
  return delta_t
```

(c)
```
// gather_nbrs: IN_NBRS
gather(D_u, D_uv, D_v):
  // update the current of the synapses
  D_uv.advance(D_v)
  return D_uv.current

sum(a,b): return a + b

apply(D_u, acc):
  // update the membrane potential
  // after getting the input current
  D_u.advance(t, acc)

// scatter_nbrs: ALL_NBRS
scatter(D_u, D_uv, D_v):
  // if D_u generate outputs
  if(D_u.genOutputs(t)):
    Activate(v)
  return delta_t
```

Fig. 3 Example codes for the base class of neuron, the base class of synapse (**a**), Q_{w} for $WA^{(r)}$ (**b**) and Q_{o} for $OA^{(r)}$ (**c**)

where NE_i is the amount of neurons in the ith region, and M is the amount of computing nodes. Similarly, the distribution of synapses placed in different nodes is calculated as:

$$\mathrm{MSY}_{\mathrm{avg}} = \frac{\sum_i \sum_j \mathrm{NE}_i \times M_{ij} \times \frac{M-1}{M}}{M}, \quad (2)$$

For the simulation of a mouse brain on the blade cluster, $\mathrm{MNE}_{\mathrm{avg}} = 4.43 \times 10^6$ and $\mathrm{MSY}_{\mathrm{avg}} = 1.16 \times 10^{10}$.

Since the distributed synapses have an important effect on the communication time during simulation, we proposed the region-based distribution method and the cluster-based distribution method in order to reduce the communication. These methods considered the fact that synaptic connections within the same region are more commonly used compared to the ones between two different regions [28]. For the region-based method, the simulation randomly placed the neurons in the same region to the same computing node. The distribution runs several times until the best result (considering both the maximum amount of neurons on one node and the average distributed synapses) is achieved. The resulting distribution by using this method on the blade cluster is shown in Fig. 4. However, the region-based method ignores the connectome information which describes the regional topology of the brain network. In order to place the closely connected regions to the same node (to further reduce the communication among computing nodes), we propose the cluster-based distribution method, where the K-means cluster algorithm is applied on the connectome to get M clusters, and the regions in the same cluster are placed in the same node. The resulting distribution by using this method on the blade cluster is shown in Fig. 5. As the plot shown in Fig. 5b, the amount of distributed synapses in each node is significantly reduced.

Finally, the mouse brain network model is with 7.11×10^7 neurons and 1.97×10^{12} synapses. It can be compared to the real mouse brain with approximately 7.11×10^7 neurons and 10^{12} synapses [5]. Simulation of 1-s biological activity of the mouse brain on the PBS platform is shown in Fig. 6a. The information processing routing in the brain starts with a visual stimulus on the retina. As illustrated in the figure, regions related to vision (VISp), memory (ECT, ENTi, CA3, SUBd), decision making (CL) and motor control (Mop) are activated in a sequential manner. The activities are shown by the fire rate change over time in the corresponding regions in Fig. 6b. In addition, the fire rates of neurons were among 15–22 Hz in this simulation, compared to a beta wave of 13–30 Hz in the recognition process of a real brain.

Human Hippocampus Network Modeling

The human hippocampus network consists of 4 major regions, including dentate gyrus (DG), cornu ammonis region 3 (CA3), cornu ammonis region 1 (CA1) and subiculum (Sub), and accepts the inputs from lateral entorhinal cortex (LEC) and medial entorhinal cortex (MEC) [8]. In the human hippocampus, there are two information processing pathways: One is from LEC and MEC to DG, then to CA3 and CA1 and finally to Sub. This pathway is mostly used for the process of memorization. The other pathway is from LEC and MEC directly to CA3 and then to CA1 and Sub. This pathway is for the process of recall [5, 31].

In order to build the hippocampus network, we obtained the connectome and the amounts of neurons per region by using the method similar to the mouse brain network. The details of the hippocampus micro-circuit and the shape of cells as well as their dendrites are also considered in the simulation. The NNA of the human hippocampus is shown in Fig. 7. The types of neurons considered in this simulation include pyramidal neuron, granule neuron, basket neuron, star neuron and common neuron, where pyramidal

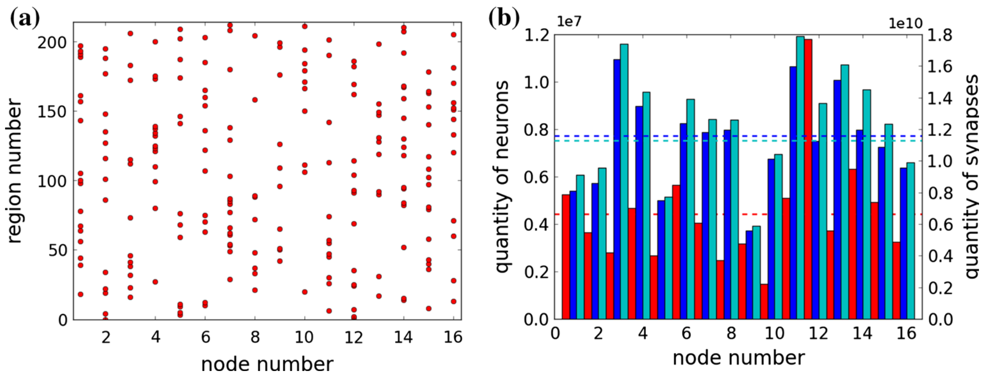
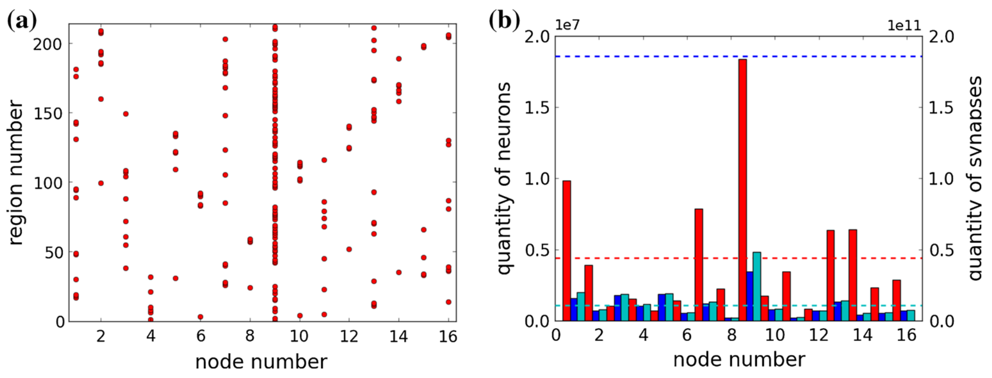


Fig. 4 Region-based distribution method. **a** Region distribution. The *dots* plot the distribution of the regions on the computing nodes. **b** Neuron and synapse distribution. The *bars* plot the number of neurons (*red*), distributed synapses (*blue*) and all related synapses (*cyan*) in each node. The *dotted lines* show the $\mathrm{MNE}_{\mathrm{avg}}$(*red*), $\mathrm{MSY}_{\mathrm{avg}}$ (*blue*) and the mean quantity of distributed synapses by using this method (*cyan*) (Color figure online)

Fig. 5 Cluster-based distribution method. Symbols are with the same meaning as used in Fig. 4



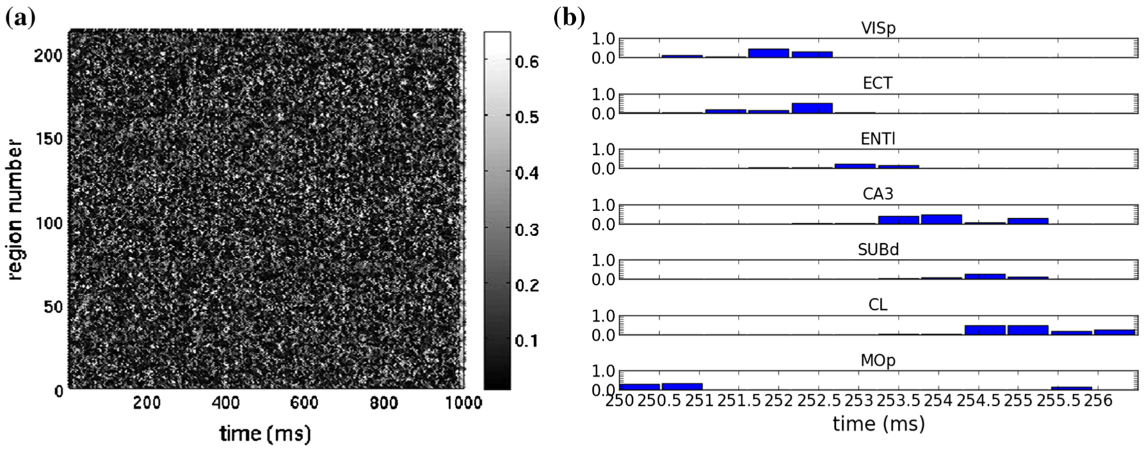


Fig. 6 Simulation results of the mouse brain network. **a** The fire rate of each region at the time interval [0, 1000]. The fire rate is represented by the *gray* level. **b** The recognition routing. Each row plots the fire rate of the regions during simulation

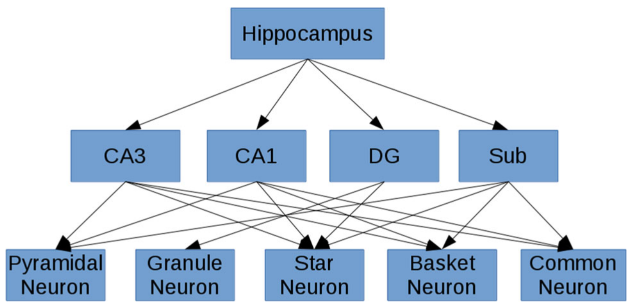


Fig. 7 The NNA for the simulated human hippocampus neural network

neurons and granule neurons are excitatory neurons (90 % of the neurons are of this type in the simulation), while basket neurons, star neurons and neurons with other shapes (common neurons) are inhibitory neurons (10 % of the neurons are of this type in the simulation). In order to simulate these neurons with different shapes, every neuron is randomly given a 3D coordinate for representing its position, denoted as (x, y, z). Each neuron accepts from and issues to the neurons in a specific position according to its shape, while the common neurons are randomly connected. The standard synapse model is used in this simulation, and the connection probabilities between any two neurons are set to 0.1 if they are in the same region, and to 0.004 if not [22].

Since there are only four brain regions (less than the amount of computing nodes in the blade cluster) considered in the human hippocampus network, both the region-based and the cluster-based distribution method cannot be used directly. Thus, we propose a region-neuron distribution method. In this method, the neurons of a specific region are randomly placed on the computing nodes related to this region. The application of this method is shown in Fig. 8. The DG (8.80×10^6 neurons, approximately 51 % of the total modeling neurons) stores on the first 8th computing nodes, the CA3 (2.30×10^6 neurons, approximately 13 % of total neurons) stores on node 9 and 10, the CA1 (4.70×10^6 neurons, approximately 27 % of total neurons) stores on node 11, 12, 13 and 14, and the CA3 (1.40×10^6 neurons, approximately 9 % of total neurons) stores on the last two nodes (the number of neurons distributed in different regions is based on [31]). The random distribution method still works in this simulation.

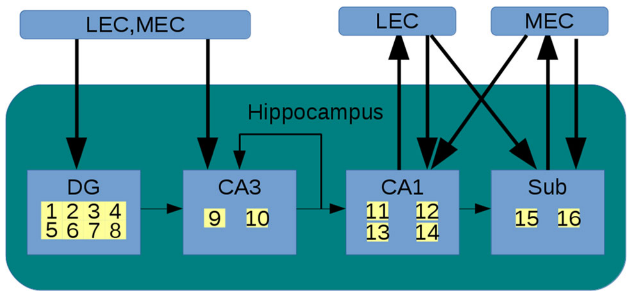


Fig. 8 The region-neuron distribution method. The regions are plot as the *blue squares*. The neurons of the region were randomly placed on the computing nodes related to this region. The *yellow squares* within each region represent the corresponding computing nodes (Color figure online)

The simulated human hippocampus neural network contains 1.72×10^7 neurons and 7.15×10^{11} synapses, compared to real human hippocampus with 1.72×10^7 neurons and approximately 10^{11} synapses [31]. The results of the simulation are shown in Fig. 9. The memorization pathway and the recall pathway of the hippocampus are shown by the region's fire rate change over time. Besides, the fire rates of neurons were among 15–22 Hz in this simulation, compared to a beta wave of 13–30 Hz in the process of human hippocampus.

Modeling CASIA Brain: A Functioning Brain-Inspired Cognitive System

The CASIA brain[7] is a functioning multi-scale brain-inspired cognitive system developed at the Institute of Automation, Chinese Academy of Sciences. It is designed to be brain-inspired from information processing mechanism perspective and human-like from cognitive functions perspective. CASIA brain 0.1.0 is composed of several functional components which are implemented by brain-inspired neural networks, including vision, encoder, decoder, working memory, long-term memory, recognition component, reasoning component and decision component, where each component simulates functions of one or several brain regions in high levels. The architecture of the CASIA brain is shown in Fig. 10a, and the corresponding NNA is shown in Fig. 10b. The NNA includes four levels of details and three types of neurons. The third layer from the top is the functional column layer implementing the functions, such as addition, multiplication, integration and convolution, by exploiting the Neural Engineering Framework (NEF) and the Echo State Networks (ESN) [10, 11, 20], and the layer above is the component layer where each functional component is composed of various functional columns. Three types of neurons are used, including the artificial neurons implementing the receptive fields used in the vision component, the artificial neurons for recording values used in the memory component and the Izhikevich spiking neurons [19] used in other components.

The CASIA brain is designed to be a general platform, on which any types of tasks can be run. We will introduce three cognitive tasks to illustrate the ability of CASIA brain, including the serial memory task, the reasoning task and the reasoning-based recognition task. To fulfill the serial memory task, the CASIA brain recognizes and memorizes a list of images and stores the results in its working memory. Whether these images were memorized successfully can be tested by asking questions, such as what was the object in a given position of the list? The serial memory task is shown in Fig. 11. In our experiments, the CASIA brain can recall a list of 13 objects at most, but the MSE (mean squared error) between the brain's output and the correct object's representation, and MSE between the output and the other objects' representation are too close to be easily distinguished when the list contains more than 9 images. In other words, the working memory of the CASIA brain can memorize about 9 objects, compared to that of human beings which recall about 7 ± 2 objects [26].

The aim of the reasoning task is to illustrate and test the reasoning ability of the CASIA brain, as shown in Fig. 12. To fulfill this task, we firstly give an image list to the brain as the input (e.g., bicycle, car and train shown in the first row), and the CASIA brain finds out that the input is a vehicles list (in the same category) and is sorted by size. Then, the brain gets the second list as shown in the second row, which further strengthens the hypothesis. The third row presents the reasoning task. Based on the analysis and understanding of the first and the second row on how they sort pictures, the CASIA brain achieves the result by searching the memory for "animal" which is "bigger" than cat and horse. The objects in the pictures such as the bicycle, the cat, and the categories such as animals and vehicles, were all represented as a float vector transmitted by firing neuron groups in the CASIA brain, and the property of the sorted list is implemented by the addition column. The third task is the reasoning-based recognition task, as shown in Fig. 13. With the help of the recognition results of the category recognition component and the reasoning component, the CASIA brain recognizes that there is a ship in this picture instead of a car.

The CASIA brain is currently used on a series of cognitive robotics named BrainBo.[8] The CASIA brain coordinates BrainBo's various cognitive behaviors, including but not limited to multi-modal sensation and perception (vision, audition, touch, etc.), language, prediction, reasoning, decision making, etc. Whenever a BrainBo is performing a cognitive task, multiple regions

[7] http://bii.ia.ac.cn/CASIA-Brain.htm.

[8] BrainBo: http://bii.ia.ac.cn/brainbo.html.

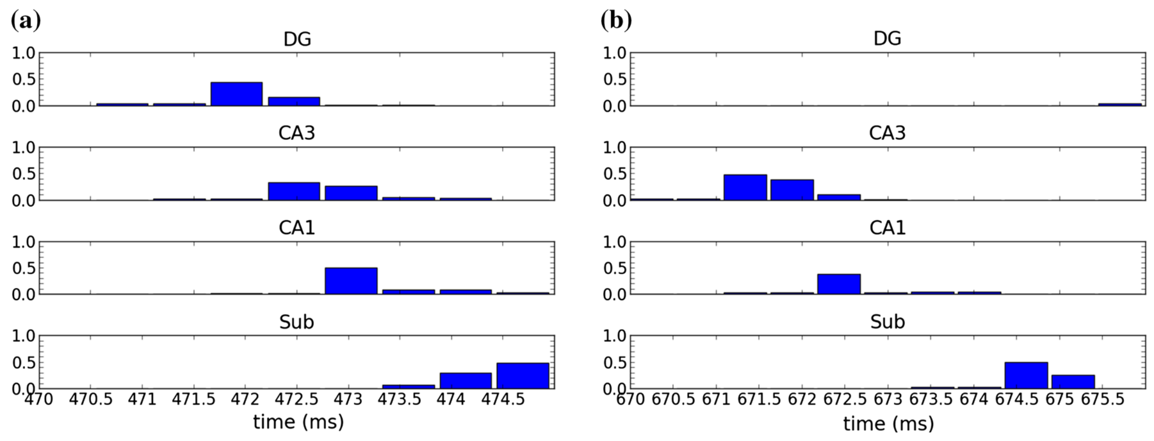


Fig. 9 Simulation results of the hippocampus network. Each row plots the firing rates of the regions during simulation. **a** The memorization procedure. **b** The recall procedure

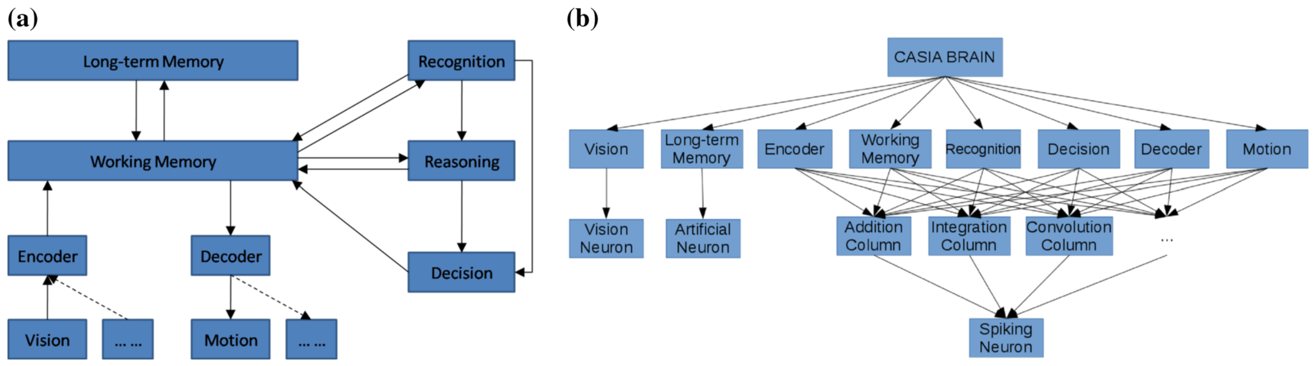


Fig. 10 CASIA brain 0.1.0. **a** The architecture of the CASIA brain 0.1.0. The brain regions list in the parentheses is the corresponding functional brain regions that the component simulates. **b** The NNA of the CASIA brain 0.1.0

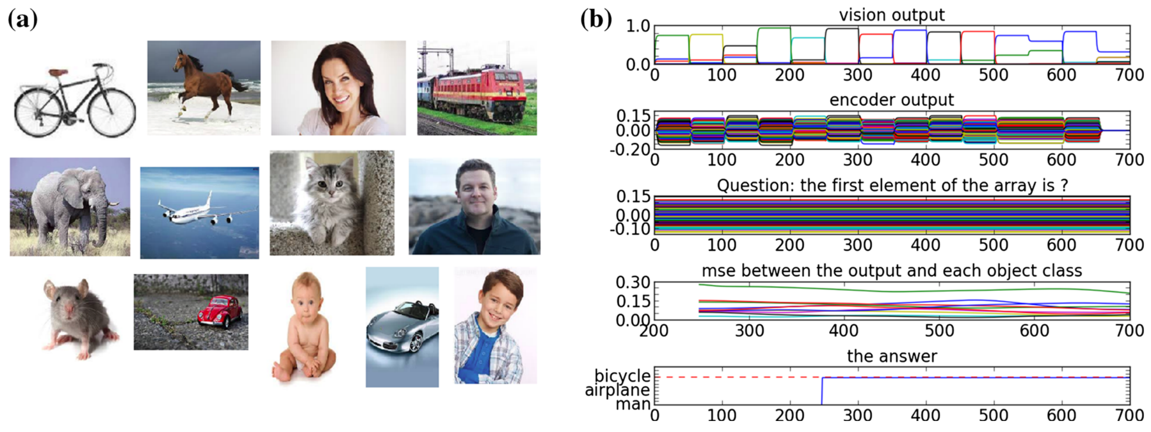


Fig. 11 The serial memory task. **a** A list of images used in the task. **b** The *first* and the *second rows* plot the outputs of the vision and encoder components of the CASIA brain. The *third row* is the output of the question column. The *fourth row* is the result on the MSE of the brain output and each object class. The *final row* plots the answer from the brain

of the CASIA brain in BrainBo are coordinating with each other to fulfill the task. Now, BrainBo can interact with human and learn basic behaviors through autonomous learning, recognize various objects from interactions with the environment and perform inductive reasoning task with the CASIA brain.

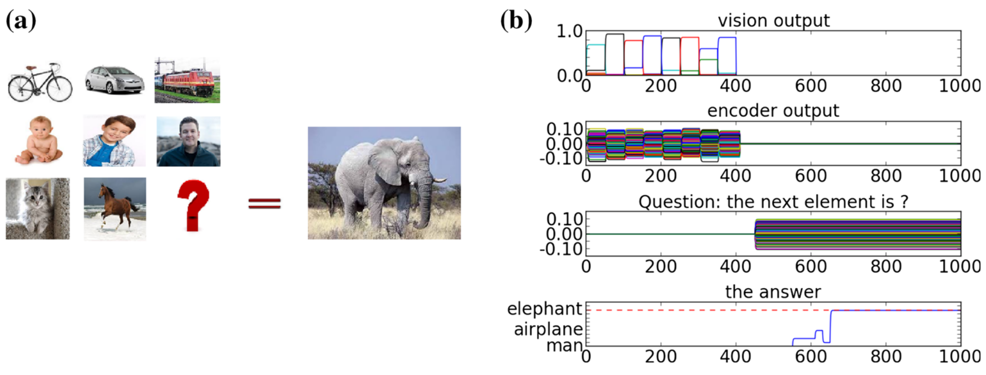


Fig. 12 The reasoning task. **a** *Two rows* of input pictures and the row including the question. **b** The *first* and the *second rows* plot the outputs of the vision and encoder components of the CASIA brain. The *third row* plots the output of the question column. The *final row* plots the answer from the CASIA brain

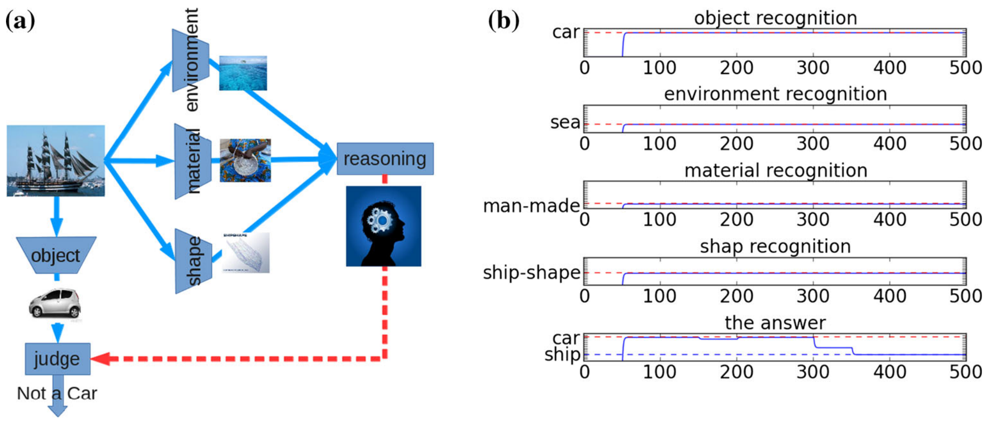


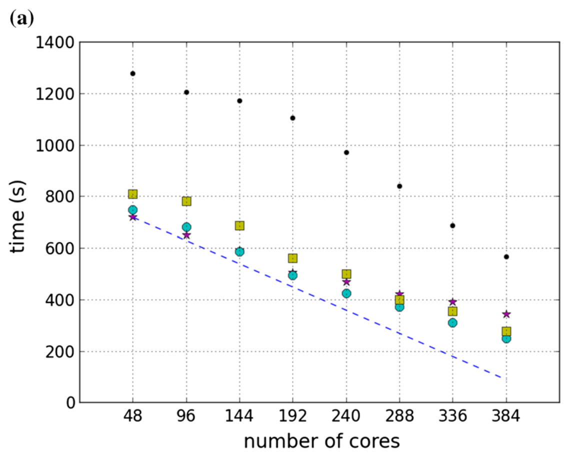
Fig. 13 The reasoning-based recognition task. **a** The task processing procedure. **b** The *first row* plots the output of the object recognition component. The next *three rows* plot the outputs of the category recognition components, including the environment component, the material component and the shape component. The final row plots the answer from the CASIA brain

Scalability and Performance

To benchmark the PBS platform, we take two primary measurements: the size of the network that could be represented in varying amounts of memory, and the time it takes to simulate one second of the modeling time. Two simulations, the lumped synapse-based mouse brain network proposed in “Mouse Whole-Brain Simulation” section and the standard synapse-based human hippocampus network in “Human Hippocampus Network Modeling” section, are used as the benchmark networks.

In order to evaluate whether the simulation makes good use of the parallel architecture of the distributed systems, a series of evaluations have been made. When keeping the number of neurons as a constant while increasing the number of used processor cores, strong scaling of simulation time is observed through evaluations on the simulated mouse brain network in Fig. 14a and the hippocampus network in Fig. 14b. As shown in Fig. 14a, for the network with a fixed size of about 3.42×10^6 neurons, the experiment reveals the high degree of parallelization of the simulation code on lumped synapse-based simulations, resulting in the favorable strong scaling. Figure 14b shows the corresponding strong scaling on the simulations with the non-lumped synapses. The size of the maximum filling network at 48 cores is about 3.75×10^5 here, ten times smaller than the lumped synapse one, due to the memory consumption of storing the non-lumped synapses. In addition, the region-based distribution method and the region-neuron-based distribution method reduce the simulation time effectively, especially on the larger clusters where the amount of neurons and synapses per node are much smaller. The decrease of neurons and synapses reduces the computational burden per node and makes the communication the most time-demanding step during simulation. Hence, the distribution methods which try to minimize the communication time contribute significantly to the reduction of total simulation time. This also explains why the cluster-based method becomes more effective as the number of machines in the cluster increases.

For the simulations of brain-inspired networks, the feasibility of a particular simulation is determined by memory constraints rather than by required performance of computing systems. Thus, we provide a maximum filling scaling investigation [21]: For a given problem size (let



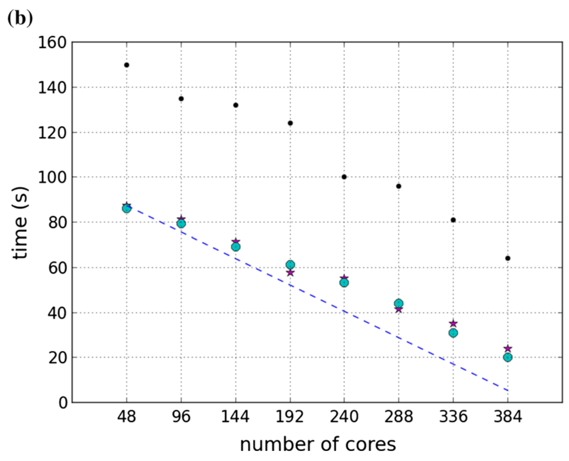
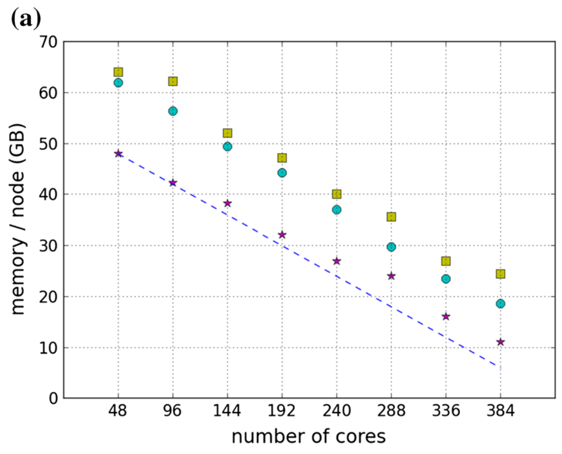


Fig. 14 Strong scaling. **a** A network of $N = 3.42 \times 10^6$ neurons is simulated on the blade cluster for the activities of 1 s of the real biological system. Optimal linear scaling is shown by the *dashed line*. The *solid dots* plot the computational time including both the initialization time and the simulation time. At 48 cores the network consumes all available memory, which leads to the highest possible workload per core for the given network size. The scaling of the simulation time with random distribution method (represented as asterisks) between 48 and 384 cores has a slope slightly below linear scaling. The simulation time with the region-based method and the cluster-based method is shown by the *dotted* symbols and the *square* symbols, respectively. **b** Strong scaling for a network of $N = 3.75 \times 10^5$ neurons on the blade cluster. The simulation time with the random method and the region-neuron-based method is shown by the *asterisk* symbols and the *dotted* symbols, respectively



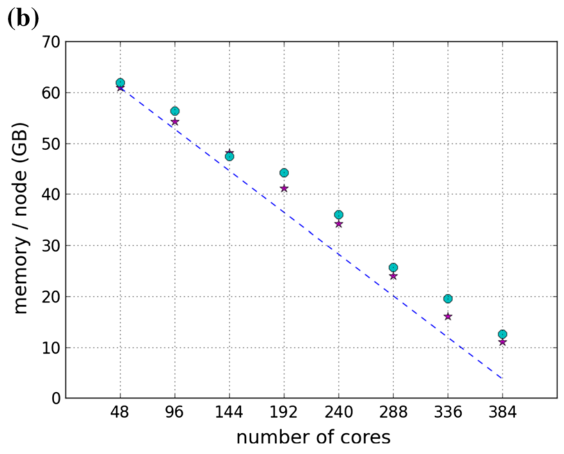
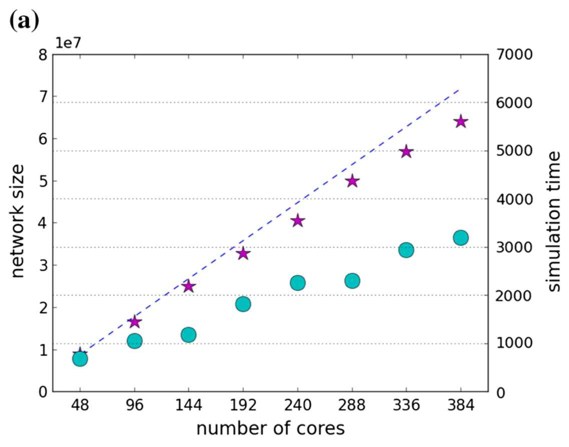


Fig. 15 The maximum filling scaling investigation. The network size (number of neurons) is chosen so that the simulation consumes all available memory. The number of neurons that are handled by one core decreases as the number of cores increases. The *dotted lines* give linear fits to the data. **a** The lumped synapse-based simulation. The memory consumption with the random distribution method, the region-based method and the cluster-based method at run-time of the simulation is shown by the *asterisk* symbols, the *dotted* symbols and the *square* symbols, respectively. **b** The standard synapse-based simulation. The memory consumption with the random distribution method and the region-neuron-based method at run-time of the simulation is shown by the *asterisk* symbols and the *dotted* symbols

N represents the number of neurons) we use the smallest portion of the computer that has sufficient memory to fit the requirements of the simulation. The memory consumption per node as a function of the size of the computing nodes decreases slightly sublinearly, as shown in Fig. 15a, b. However, due to the imbalance distribution of neurons and synapses, with the region-based method or the cluster-based method, the node with maximum memory consumption consumes more memory than that with the random method. Hence, the maximum network size with these methods is smaller than with the random method. Figure 16 shows the results of the weak scaling investigation, where the problem size increases with the increasing amount of memory. The plots illustrate a very good weak scaling in terms of memory, since it produces a sublinear slope. By exploiting the fat cluster, we have implemented a



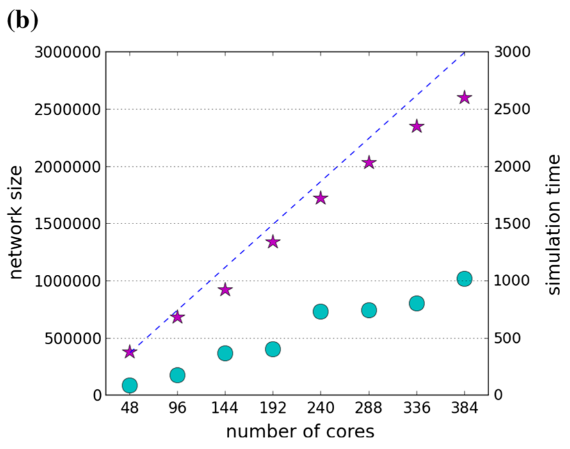


Fig. 16 The weak scaling investigation. The total size of the network with the random distribution method (i.e., number of neurons) is a function as the number of cores changes (represented with *asterisk* symbols). The optimal linear scaling is represented as *dashed lines*. The function related to the *asterisk* symbols gives near-linear fits to the data. The corresponding time required for simulating 1 s of the activity for biological system is shown as the *dotted* symbols. **a** The lumped synapse-based simulation. **b** The standard synapse-based simulation

Table 1 Performance measures of the PBS platform for 384 cores simulating lumped synapse (left column) and standard synapse (right column)-based networks on the blade cluster, obtained with the gprof profiling tool

	Lumped synapse model	Standard synapse model
MFLOPS	112.334	88.229
MFLOPS/PEAK (%)	0.528	0.327
MIPS	1659	821
MIPS/PEAK (%)	12.579	9.263

lumped synapse-based network with 2.11×10^8 neurons and 5.91×10^{11} synapses, and a standard synapse-based network with 1.01×10^7 neurons and 3.72×10^{10} synapses.

In order to measure the performance of PBS on the clusters, the gprof profiling tool[9] is used. We simulate the lumped synapse-based network with 7.11×10^7 neurons and 1.97×10^{12} synapses, and the standard synapse-based network with 3.11×10^6 neurons and 1.24×10^{11} synapses on the blade cluster, and the number of floating point operations per second per core (MFLOPS) and instructions per second per core (MIPS) is obtained, as shown in Table 1. Since the Izhikevich spiking neuron model is used, the floating point performance is thus expected to be low. The MIPS performance is good, especially considering that the simulation heavily relies on the random memory access. Besides, the lumped synapse-based simulation shows a higher performance on MFLOPS and MIPS, since the communication decreases through using simpler synapse model.

9 http://sourceware.org/binutils/docs/gprof/.

Discussions and Conclusions

By exploiting the graph computation technique, we have built the parallel and distributed platform for modeling and simulating the cognitive brain. Instead of using modeling method with the Python scripts, PBS provides an extended-DAG interface, which is a more complicated but arguably expressive modeling method for the modeling of brain-inspired networks at multiple scales. However, in order to facilitate users to build their own models, we also provide an interface with Python scripts as an option. Our platform is evaluated on two clusters, which achieves good scalability and performance. These experiments show attractive strong/weak scaling behavior of our simulations, and hence, better and faster supercomputers will certainly reduce the simulation time and increase the maximum size of the networks. In the future, we would like to optimize the memory usage during simulation and to explore the distribution method by considering more cluster details such as the racks and the bandwidth.

Several neural networks have been built and simulated on PBS, and three of them are introduced in this paper. By exploiting the CASIA cluster with 512 cores and 3 TB memory, a brain-inspired neural network with 2.11×10^8 neurons and 5.91×10^{11} synapses is modeled and simulated on this platform. In the future, we would like to refine the human hippocampus network and the mouse whole-brain network by adding more biological details and use these models to test large-scale hypotheses of the brain. We are also developing real-world applications with various cognitive functions based on the CASIA brain and deploy them on various robots, which will be reported in the near future.

Funding This study was funded by the Strategic Priority Research Program of the Chinese Academy of Sciences (XDB02060007) and Beijing Municipal Science and Technology Commission (Z151100000915070).

Compliance with Ethical Standards

Conflict of Interest Xin Liu, Yi Zeng, Tielin Zhang, and Bo Xu declare that they have no conflict of interest.

Informed Consent All procedures followed were in accordance with the ethical standards of the responsible committee on human experimentation (institutional and national) and with the Helsinki Declaration of 1975, as revised in 2008 (5). Additional informed consent was obtained from all patients for which identifying information is included in this article.

Human and Animal Rights This article does not contain any studies with human or animal subjects performed by the any of the authors.

References

1. Aleksander I, Morton H. An introduction to neural computing. London: Chapman and Hall; 1990.
2. Ananthanarayanan R, Esser S, Simon H, Modha D. The cat is out of the bag: cortical simulations with 109 neurons and 1013 synapses. In: Proceedings of the SC conference on high performance networking and computing, 2009. 2009;1: p. 1–12 IEEE.
3. Azevedo F, Carvalho L, Grinberg L, Farfel JM, Ferretti R, Leite R, Jilho WJ, Lent R, Herculano-Houzel S. Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain. J Comp Neurol. 2009;513(5):532–41.
4. Beeman D, Wang Z, Edwards M, Bhalla U, Cornelis H, Bower JM. The GENESIS 3.0 project: a universal graphical user interface and database for research, collaboration, and education in computational neuroscience. BMC Neurosci. 2007;8(suppl 2):4.
5. Boss BD, Peterson GM, Cowan WM. On the number of neurons in the dentate gyrus of the rat. Brain Res. 1985;338(1):144–50.
6. Brette R, Rudolph M, Carnevale T, Hines M, Beeman D, Bower JM, Diesmann M, Morrison A, Goodman PH, Zirpe M, Natschlger T, Pecevski D, Ermentrout B, Djurfeldt M, Lansner A, Rochel O, Vieville T, Muller E, Davison AP, Boustani SE, Destexh A. Simulation of networks of spiking neurons: a review of tools and strategies. J Comp Neurosci. 2007;23(3):349–98.
7. Carnevale N, Hines M. The NEURON book. Cambridge: Cambridge University Press; 2006.
8. Cutsuridis V, Graham B, Cobb S, Vida I. Hippocampal microcircuits. New York: Springer; 2010.
9. Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters. In: Proceedings of the USENIX symposium on operating systems design and implementation, 2004. OSDI, 2004. 2004; IEEE.
10. Eliasmith C. How to build a brain: a neural architecture for biological cognition. Oxford: Oxford University Press; 2013.
11. Eliasmith C, Anderson C. Neural engineering: computation, representation, and dynamics in neurobiological systems. Cambridge: MIT press; 2004.
12. Eliasmith C, Stewart T, Choo X, Bekolay T, Dewolf T, Tang Y, Rasmussen D. A large-scale model of the functioning brain. Science. 2012;338:1202–5.
13. Eliasmith C, Trujilo O. The use and abuse of large-scale brain models. Curr Option Neurobiol. 2014;25:1–6.
14. Gewaltig MO, Morrison A, Plesser HE. NEST by example: an introduction to the Neural Simulation Tool NEST. In: Le Novère N, editor. Computational systems neurobiology. Springer; 2012. p. 533–558.
15. Gonzalez J, Low Y, Gu H, Bickson D. Powergraph: distributed graph-parallel computation on natural graphs. In: Proceedings of the USENIX symposium on operating systems design and implementation, 2012. OSDI, 2012. 2012; IEEE.
16. Goodman D, Brette R. Brian: a simulator for spiking neural networks in python. Front Neuroinform. 2008;2:5.
17. Hammarlund P, Ekeberg O. Large neural network simulations on multiple hardware platforms. J Comput Neurosci. 1998;5(4):443–59.
18. Hodgkin AL, Huxley AF. A quantitative description of membrane current and its application to conduction and excitation in nerve. J Physiol. 1952;117(4):500–544.
19. Izhikevich E. Simple model of spiking neurons. IEEE Trans Neural Netw. 2003;14:1569–72.
20. Jaeger H. The “echo state” approach to analysing and training recurrent neural networks-with an erratum note. In: Technical report, German National Research Center for Information Technology. 2001.
21. Kunkel S, Schmidt M, Eppler JM, Plesser HE, Masumoto G, Igarashi J, Ishii S, Fukai T, Morrison A, Diesmann M, Helias M. Spiking network simulation code for petascale computers. Front Neuroinform. 2014;10(8):78.
22. Lansner A, Diesmann M. Virtues, pitfalls, and methodology of neuronal network modeling and simulations on supercomputers. In: Proceedings of the Chapter 10 in Nicolas Le Novre computational systems biology, Springer 2012.
23. Le NN. Computational systems neurobiology. New York: Springer; 2012.
24. Low Y, Gonzalez J, Kyrola A, Bickson D, Guestrin C, Hellerstein JM. Distributed graphlab: a framework for machine learning and data mining in the cloud. In: Proceedings of the very large database endowment, 2012. VLDB, 2012. 2012;5(8), IEEE.
25. Malewicz G, Austern MH, Bik AJC, Dehnert JC, Horn I, Leiser N, Czajkowski G. Pregel: a system for large-scale graph processing. In: ACM special interest group on management of data, 2010. SIGMOD, 2010. 2010; IEEE.
26. Miller G. The magical number seven, plus or minus two: Some limits on our capacity for processing information. Psychol Rev. 1956;63(2):81–97.
27. Morrison A, Aertsen A, Diesmann M. Spike-timing dependent plasticity in balanced random networks. Neural Comput. 2007;19: 1437–67.
28. Oh SW, Harris JA, Ng L, Winslow B, Cain N, Mihalas S, Wang Q, Lau C, Kuan L, Henry AM, Mortrud MT, Ouellette B, Nguyen TN, Sorensen SA, Slaughterbeck CR, Wakeman W, Li Y, Feng D, Ho A, Nicholas E, Hirokawa KE, Bohn P, Joines KM, Peng H, Hawrylycz MJ, Phillips JW, Hohmann JG, Wohnoutka P, Gerfen CR, Koch C, Bernard A, Dang C, Jones AR, Zeng H. A mesoscale connectome of the mouse brain. Nature. 2014;508:207–14.
29. Pecevski D, Natschlger T, Schuch K. Pcsim: a parallel simulation environment for neural circuits fully integrated with python. Front Neuroinform. 2009;3:11.
30. Rotter S, Diesman M. Exact digital simulation of time-invariant linear systems with applications to neuronal modeling. Biol Cybern. 1999;81(5/6):381–402.
31. Seress L. Interspecies comparison of the hippocampal formation shows increased emphasis on the region superior in the ammon’s horn of the human brain. J Hirnforsch. 1988;29:335–40.
32. Song S, Miller KD, Abbott LF. Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. Nat Neurosci. 2000;3:919–26.

33. Yonezawa A, Watanabe T, Yokokawa M, Sato M, Hirao K. Advanced institute for computational science (aics): Japanese national high-performance computing research institute and its 10-petaflops supercomputer. In: Proceedings of the SC conference on high performance networking and computing, 2011 2011.13, pp. 1–8 IEEE.

34. Zhang T, Zeng Y, Xu B. A computational effort towards the microscale mouse brain connectome from the mesoscale. Manuscript, 2015.