frontiers in Cellular Neuroscience

ORIGINAL RESEARCH
published: 07 April 2021
doi: 10.3389/fncel.2021.623552

Real-Time Simulation of a Cerebellar Scaffold Model on Graphics Processing Units

Rin Kuriyama[1], *Claudia Casellato*[2], *Egidio D'Angelo*[2,3] *and Tadashi Yamazaki*[1]*

[1] *Graduate School of Informatics and Engineering, The University of Electro-Communications, Tokyo, Japan,*
[2] *Neurophysiology Unit, Neurocomputational Laboratory, Department of Brain and Behavioral Sciences, University of Pavia, Pavia, Italy,* [3] *IRCCS Mondino Foundation, Pavia, Italy*

Large-scale simulation of detailed computational models of neuronal microcircuits plays a prominent role in reproducing and predicting the dynamics of the microcircuits. To reconstruct a microcircuit, one must choose neuron and synapse models, placements, connectivity, and numerical simulation methods according to anatomical and physiological constraints. For reconstruction and refinement, it is useful to be able to replace one module easily while leaving the others as they are. One way to achieve this is via a scaffolding approach, in which a simulation code is built on independent modules for placements, connections, and network simulations. Owing to the modularity of functions, this approach enables researchers to improve the performance of the entire simulation by simply replacing a problematic module with an improved one. Casali et al. (2019) developed a spiking network model of the cerebellar microcircuit using this approach, and while it reproduces electrophysiological properties of cerebellar neurons, it takes too much computational time. Here, we followed this scaffolding approach and replaced the simulation module with an accelerated version on graphics processing units (GPUs). Our cerebellar scaffold model ran roughly 100 times faster than the original version. In fact, our model is able to run faster than real time, with good weak and strong scaling properties. To demonstrate an application of real-time simulation, we implemented synaptic plasticity mechanisms at parallel fiber–Purkinje cell synapses, and carried out simulation of behavioral experiments known as gain adaptation of optokinetic response. We confirmed that the computer simulation reproduced experimental findings while being completed in real time. Actually, a computer simulation for 2 s of the biological time completed within 750 ms. These results suggest that the scaffolding approach is a promising concept for gradual development and refactoring of simulation codes for large-scale elaborate microcircuits. Moreover, a real-time version of the cerebellar scaffold model, which is enabled by parallel computing technology owing to GPUs, may be useful for large-scale simulations and engineering applications that require real-time signal processing and motor control.

Keywords: cerebellum, spiking network, graphics processing unit, real-time simulation, scaffolding approach

OPEN ACCESS

Edited by:
Tycho Hoogland,
Erasmus Medical Center, Netherlands

Reviewed by:
Robert Andrew McDougal,
Yale University, United States
Pablo Varona,
Autonomous University of Madrid, Spain

***Correspondence:**
Tadashi Yamazaki
fcns20@neuralgorithm.org

Specialty section:
This article was submitted to Cellular Neurophysiology, a section of the journal Frontiers in Cellular Neuroscience

Received: *30 October 2020*
Accepted: *15 March 2021*
Published: *07 April 2021*

Citation:
Kuriyama R, Casellato C, D'Angelo E and Yamazaki T (2021) Real-Time Simulation of a Cerebellar Scaffold Model on Graphics Processing Units. Front. Cell. Neurosci. 15:623552. doi: 10.3389/fncel.2021.623552

1. INTRODUCTION

Flexibility and efficiency are important factors in large-scale computer simulation of spiking neural networks (Brette et al., 2008; Eppler et al., 2009). Flexibility means that spiking neural networks can be built and updated easily and quickly. To ensure flexibility, the simulation software should comprise a number of functional modules that are independent or only loosely dependent, which in turn introduces redundancy that can reduce efficiency in terms of greater memory usage and slower computation. On the other hand, efficiency means that simulation of spiking neural networks can be carried out efficiently in terms of memory and network usage, and computational speed. For example, faster simulation allows us to study biophysical processes that take a long time for hours and days in a reasonable time. A milestone of faster simulation may be real-time simulation. Real-time simulation enables simulated models to be used for engineering applications that require realtime signal processing and motor control, including robot control (Yamazaki et al., in press). To ensure efficiency, simulation software should be written carefully while integrating everything in the code tightly to remove redundancy as much as possible, which in turn could make the code unreadable and difficult for refactoring. Thus, flexibility and efficiency are somewhat conflicting concepts in simulations. How can we compromise between these two factors?

A solution is to use dedicated simulators such as NEST (Gewaltig and Diesmann, 2007), and NEURON (Carnevale and Hines, 2006). These simulators provide integrated environments consisting of easy-to-use interfaces for flexible modeling and also optimized numerical codes for efficient simulation. In fact, these simulators have been used for various projects such as the Human Brain Project (Amunts et al., 2016) in the EU, and Brain Initiatives (Ramos et al., 2019) in the US. Nevertheless, due to large-scale collaborative development, the development of such simulators would not be fast enough to support the latest technologies in the field of high-performance computing (HPC). For example, graphics processing units (GPUs) are becoming popular as hardware for parallel computing, but the NEST simulator has not yet supported it. The NEURON simulator supports GPUs, but the developers had to first extract a module for numerical simulation (Kumbhar et al., 2019), suggesting that they took a modular approach. Therefore, an integrated approach would make the use of the latest HPC technology difficult.

In an intermediate approach known as "scaffolding" (Casali et al., 2019), simulation software is divided into a number of loosely connected or functionally independent modules for neuron and synapse models, connectivity, and numerical methods. An advantage of this approach is that any module can be replaced easily for better descriptions and performance without affecting the other modules. In fact, Casali et al. (2019) replaced simulation modules from/to PyNEST and NEURON with Python interface, and while obtained the same simulation results. Thus, the scaffolding approach is another solution to finding a compromise between flexibility and efficiency.

The present study aims to provide one more proof of concept for the scaffolding approach. In this study, we developed a simulation module that uses GPUs from scratch, replacing a cerebellar scaffold model built in a previous study (Casali et al., 2019), while reusing the other modules as they are. By replacing the simulation module only, we were able to obtain the qualitatively similar simulation results, but the simulations were accelerated by about 100 times, which resulted in faster-than-real-time simulation. Furthermore, we implemented synaptic plasticity, and conducted simulation of a behavioral experiment on eye movement reflex. These results suggest that the scaffolding approach is a promising method for large-scale spiking network simulation.

2. MATERIALS AND METHODS

2.1. Overview of the Cerebellar Scaffold Model

The cerebellar scaffold model (Casali et al., 2019) is a spiking network model of the cerebellar microcircuit built based on the scaffolding approach. According to the approach, the model consists of three modules: a cell placement module, a connectivity module, and a functional simulation module. These modules model and place neurons in a 3D space, create structural connections between pairs of neurons, and simulate dynamics of the network, respectively. The first two are written in Python, whereas the simulation module is written in PyNEST and NEURON. Owing to the modular approach, one can choose appropriate simulator. In fact, one will be able to even replace a cell placement module to use multi-compartment neuron models. The entire circuit includes 7,070 mossy fibers (MFs), 219 Golgi cells (GoCs), 88,158 granule cells (GrCs), 69 Purkinje cells (PCs), 603 stellate cells (SCs), 603 basket cells (BCs), and 12 deep cerebellar nuclei (DCNs). GrCs extend ascending axon (AA) and parallel fibers (PF). All cells are placed in a three-dimensional volume of $400 \times 400 \times 900$ μm^3 (**Figure 1A**). The neurons are connected based on known anatomy of the cerebellar circuit (**Figure 1B**). The original model assumed that glomeruli provide excitatory inputs to DCN, GrC, and Goc, whereas in the present study, we replaced those glomeruli with MFs based on the known anatomy of the cerebellar circuit (Eccles et al., 1967; Ito, 1984).

MFs are modeled as Poisson spike generators. All neurons are modeled as conductance-based leaky integrate-and-fire (LIF) units, and synapses are modeled as conductance-based exponential decay synapses.

A LIF model is defined as follows:

$$\begin{aligned} C_{\mathrm{m}} \frac{\mathrm{d}u_i(t)}{\mathrm{d}t} &= -g_{\mathrm{L}}(u_i(t) - E_{\mathrm{L}}) + I_{\mathrm{e}} + I_{\mathrm{syn}}(u_i(t), t) \\ t_i^{(f)} : u_i\left(t_i^{(f)}\right) &= V_{\mathrm{th}} \\ \lim_{t_i^{(f)} \to t, t > t_i^{(f)}} u_i(t) &= V_{\mathrm{r}} \end{aligned} \tag{1}$$

where C_{m}, u_i, t, g_{L}, E_{L}, I_{e}, I_{syn}, $t_i^{(f)}$, V_{th}, and V_{r} are the membrane capacitance, membrane potential of neuron i,

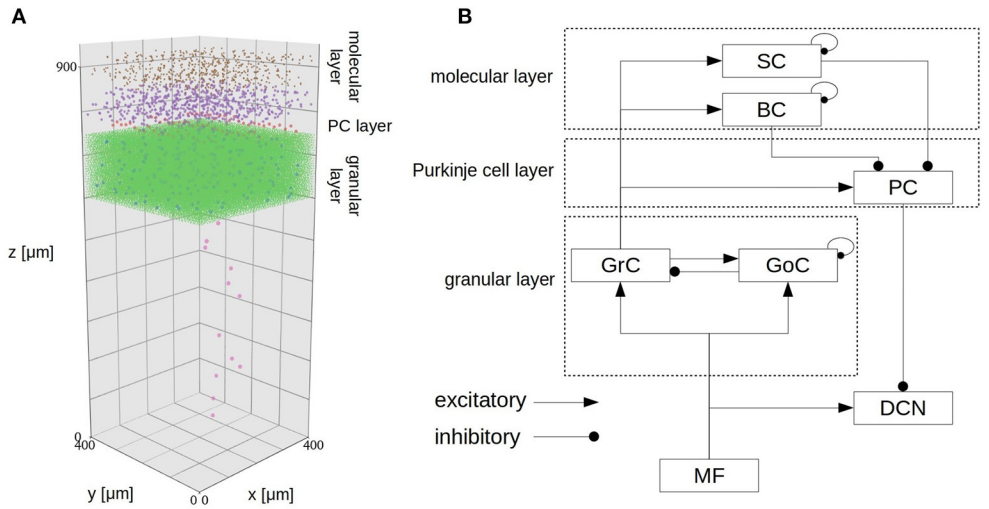

FIGURE 1 | Spatial arrangement of neurons in the cerebellar scaffold model. **(A)** Cell placement in a 3D volume of 400 × 400 × 900 μm^3. Dots represent neurons, where the colors represent cell types: MF terminal (orange), GoC (blue), GrC (green), PC (red), SC (purple), BC (brown), and DCN (pink). The entire volume contains 96,737 neurons and 4,220,752 synapses. We transposed the z axis to the vertical direction for consistency with the standard atlas. The positions were given by the cell placement module of the original scaffold model (Casali et al., 2019). **(B)** Schematic of connectivity in the same model. Abbreviations are as in the text.

time, leak conductance, resting potential, endogenous current, synaptic current, time when neuron i fires the fth spike, threshold potential, and reset potential, respectively. The first equation describes how the membrane potential is updated. The second equation defines the firing time $t_i^{(f)}$ when the membrane potential reaches the threshold V_{th}. The third equation resets the membrane potential to the reset potential V_{r} after emitting a spike. These values are set for each neuron type (**Table 1**). A synaptic current and an exponential decay synapse are defined as follows:

$$I_{\text{syn}}\left(u_i(t),t\right) = -\sum_x g_x(t)\left(u_i(t)-E_x\right)$$
$$\tau_x \frac{\mathrm{d}g_x(t)}{\mathrm{d}t} = -g_x(t) + \sum_f \sum_j w_{ij}\delta\left(t - t_j^{(f)} - t_{\text{delay}_{ij}}\right) \quad (2)$$

where x is a synapse label representing either inhibitory (inh) or excitatory (exc), g_x, E_x, τ_x, w_{ij}, and $t_{\text{delay}_{ij}}$, are the synapse conductance, reversal potential, decay time constant, synaptic weight between the presynaptic neuron j and the postsynaptic neuron i, and synaptic delay, respectively. Parameters are set as in **Table 2**.

In the present study, we reused the cell placement module and connectivity module with the same set of parameters. We replaced only the simulation module as in section 2.3.

2.2. Parallel Computing on GPUs

Parallel computing is a way to accelerate numerical calculation by dividing a problem into a number of smaller problems and solving them in parallel. GPUs are hardware specialized for computer graphics, but can be used as parallel computing accelerators. A GPU can issue a number of computing entities called "threads" simultaneously, where these threads execute the same function with different input parameters. To date, GPUs play considerable roles in the HPC field.

TABLE 1 | Neuron-specific parameters.

Type	GrC	GoC	BC	SC	PC	DCN
N.cells	88,158	219	603	603	69	12
C_m [pF]	3	76	14.6	14.6	620	89
g_L [ms]	1.5	3.6	1.0	1.0	7.0	1.56
E_L [mV]	−74	−65	−68	−68	−62	−59
Δt_{ref} [ms]	1.5	2	1.6	1.6	0.8	3.7
I_e [pA]	0	36.8	15.6	15.6	600	55.8
V_r [mV]	−84	−75	−78	−78	−72	−69
V_{th} [mV]	−42	−55	−53	−53	−47	−48
τ_{exc} [ms]	0.5	0.5	0.64	0.64	0.5	7.1
τ_{inh} [ms]	10	15	2	2	1.6	13.6

Abbreviations are as in the text.

In this study, we used 4 NVIDIA Tesla V100 GPUs installed into a DGX Station (NVIDIA, 2020b). Briefly, a V100 GPU has 5,120 computing cores, and 16 GB of high-bandwidth memory. The peak performance is 15.7 TFLOPS in single-precision operations. More detailed information of Tesla V100 GPU architecture is described elsewhere (NVIDIA, 2017).

For GPU programming, NVIDIA provides Compute Unified Device Architecture (CUDA), which is a parallel computing platform and a programming model for GPGPU (General-Purpose computing on GPUs) (NVIDIA, 2020a). Programmers can write a code for GPUs with C/C + + and CUDA extensions. In CUDA, CPUs and their memory are called

TABLE 2 | Synaptic parameters for each connection type.

Connection type (exc/inh)	Weight [nS]	Delay [ms]
MF-GrC (exc)	9.0	4.0
MF-GoC (exc)	2.0	4.0
GoC-GrC (inh)	5.0	2.0
GoC-GoC (inh)	8.0	1.0
AA-GoC (exc)	20.0	2.0
PF-GoC (exc)	0.4	5.0
SC-SC (inh)	2.0	1.0
BC-BC (inh)	2.5	1.0
PF-SC (exc)	0.2	5.0
PF-BC (exc)	0.2	5.0
SC-PC (inh)	8.5	5.0
BC-PC (inh)	9.0	4.0
AA-PC (exc)	75.0	2.0
PF-PC (exc)	0.02	5.0
PC-DCN (inh)	0.0075	4.0
MF-DCN (exc)	0.006	4.0

Abbreviations are as in the text.

"hosts," whereas GPUs and their memory are called "devices." Functions computed on GPUs are called "kernels." Programmers define kernel functions, which are automatically executed by multiple threads with different input parameters. Moreover, although all threads compute the same kernel in principle, when a feature called "streams" is used, different kernels can be executed simultaneously. More detailed documentation is available elsewhere (NVIDIA, 2020a).

2.3. A New Implementation of the Simulation Module on GPUs

We reimplemented the simulation module of the cerebellar scaffold model, which was written in PyNEST (Eppler et al., 2009), in C language with CUDA extensions from scratch on GPUs. In the new simulation module, Equations (1) and (2) were solved numerically with a forward Euler method with a temporal resolution (Δt) of 0.1 ms, whereas exchanging spike information and performing product-sum operations for synaptic inputs (the 2nd term in the RHS of Equation 2) were made for each 1 ms. These temporal resolutions are identical with the original module. Meanwhile, the original module used the 4th-order Runge-Kutta method instead of Euler method. Euler methods seem to be sufficient for more complicated models such as Izhikevich model (Izhikevich, 2003), so do for LIF models. A flowchart of the calculations is shown in **Figure 2**.

We used GPUs to perform the above calculation in parallel. Below, we explain the case of a single GPU, and later we will explain how to use multiple GPUs. A basic strategy is to assign 1 neuron to 1 thread and calculate membrane potentials in parallel. During the calculation, the following techniques were used.

First, a connectivity matrix was stored in a compressed sparse row (CSR) format for each pair of presynaptic and postsynaptic neuron types. A CSR format is an efficient way to store sparse matrix. Some other possible formats are coordinates (COO), ELLPACK (ELL), and list of lists (Shahnaz et al., 2005). Second, to generate Poisson spikes on each thread, we used a Philox counter-based random number generator (Manssen et al., 2012). Third, we overlapped spike propagation that requires data transfer between host (CPU) and device (GPU) with calculation of neurons and synapses so as to hide the latency caused by the data transfer. This was achieved by assuming a delay of 1 ms in synaptic transmission (Hines et al., 2011; Igarashi et al., 2019). The overlap of calculation and data transfer was made by using streams in CUDA.

Furthermore, we applied parallel computing techniques to calculate synaptic inputs for each neuron. For neurons that are many and have a small number of synapses, such as granule cells, we assigned the calculation for each neuron to each thread. This might be a trivial parallelization. On the other hand, for neurons that are a few and have a large number of synapses, such as Purkinje cells, we used a bunch of threads to calculate synaptic inputs for each cell, and repeated the same calculation to enumerate all cells. Specifically, we used a tree-based approach called "parallel reduction" (Harris, 2011), which reduces product-sum operations recursively (**Figure 3**). We issued 8,192 threads to perform reduction of 29,196 PF synapses on 1 PC For other neurons, we took an intermediate approach, which was to use a few threads for each cell. More detailed explanation on which parallelization is used for each neuron type is available in **Supplementary Material**.

In addition to the above GPU-specific techniques, we also used a pthread library in a host code to asynchronously perform data transfer from device to host and file I/O to record spikes of all neurons for each k_{record} time step.

To use multiple GPUs, we split an entire network into multiple smaller subnetworks in one dimension along the transversal axis, and allocate each subnetwork to each GPU. Then, we executed the same kernel on all GPUs as in a manner of single instruction, multiple threads (SIMT) over multiple GPUs, although it is possible to execute different kernels on different GPUs. Spikes calculated on each GPU were transferred to its neighboring GPUs for further calculations. We used OpenMP to execute a kernel on multiple GPUs in parallel.

Under multi-GPU setting, DCN neurons must receive inputs from PCs on different GPUs. Because all spike information including spikes emitted by PCs are transferred from GPUs to the host for each 50 ms to generate the output data, we calculated DCN neurons on a host.

2.4. Reproducibility of Simulation Results

To examine whether our simulation code reproduces the same results with sufficient accuracy, we conducted the same simulation as in Casali et al. (2019). Briefly, we fed Poisson spikes of 1Hz tonically to all MFs for 1,000 ms as background noise. We also fed phasic Poisson spikes of 150 Hz, starting from 300 ms of the stimulation onset for 50 ms, to MFs at the center of the granular layer within the radius of 140μm. During the simulation, spike trains of all neurons were obtained for these three periods. For each period, we calculated the mean firing rates of all neuron types and their standard deviations, and compared the values

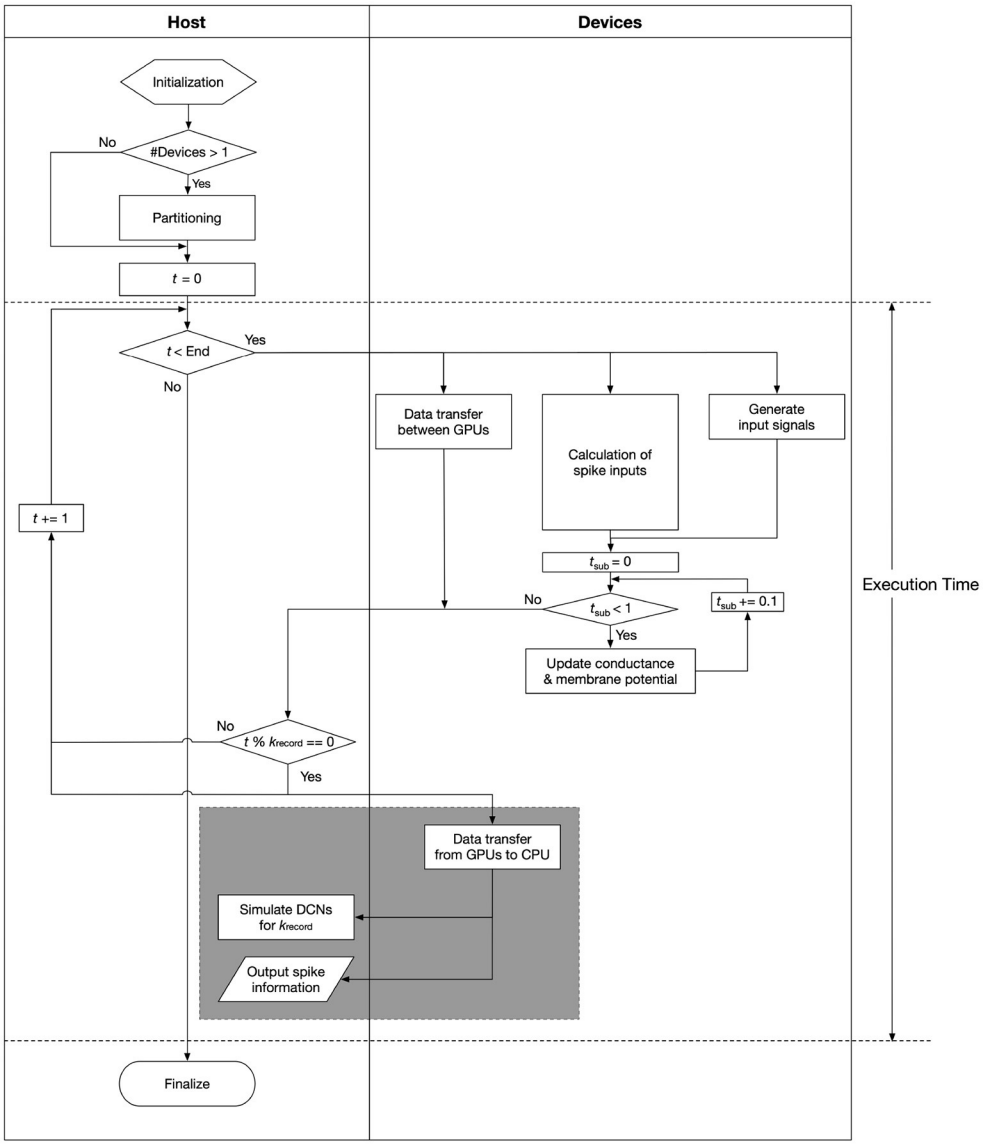

FIGURE 2 | A flowchart of the calculations. Processes in gray rectangle, which manage file I/O and simulation of DCN neurons, are performed in parallel with the rest of the numerical calculations by using pthread library and streams on CUDA. Execution time means the time spent in the area between the dashed lines.

with those obtained by the original model. For averaging, we repeated the stimulation 10 times.

2.5. Weak and Strong Scaling Properties

Weak scaling and strong scaling are two important measurements for parallel computing applications. In weak scaling, we measure computational time of a simulation while increasing the size of a model and the number of GPUs simultaneously, where the computational load for a computer node is kept the same. Good weak scaling suggests that the size of a model can be increased arbitrarily as long as enough computational resources are available. On the other hand, in strong scaling, we measure computational time while increasing the number of GPUs with a fixed model size. Good strong scaling suggests that the computer simulation can be accelerated when more computer nodes are available.

To examine the scaling, we first prepared three cerebellar scaffold models with the size of $200 \times 800 \times 900$, $400 \times 800 \times 900$, and $800 \times 800 \times 900$ μm^3, respectively. For weak scaling, we used 1, 2, and 4 GPUs to simulate the three models, respectively. For strong scaling, we performed simulation of the largest model with 1, 2, and 4 GPUs. In the simulation, spontaneous activity with 1 Hz MF spikes were fed as background noise for 10 s of the biological time. The same simulation was repeated 10 times to calculate the mean of the simulation time.

2.6. Real-Time Simulation in a More Realistic Scenario

Furthermore, we carried out simulation in a more realistic scenario, which is gain adaptation of eye movements known as optokinetic response (OKR) (Ito, 1984). OKR is an eye movement

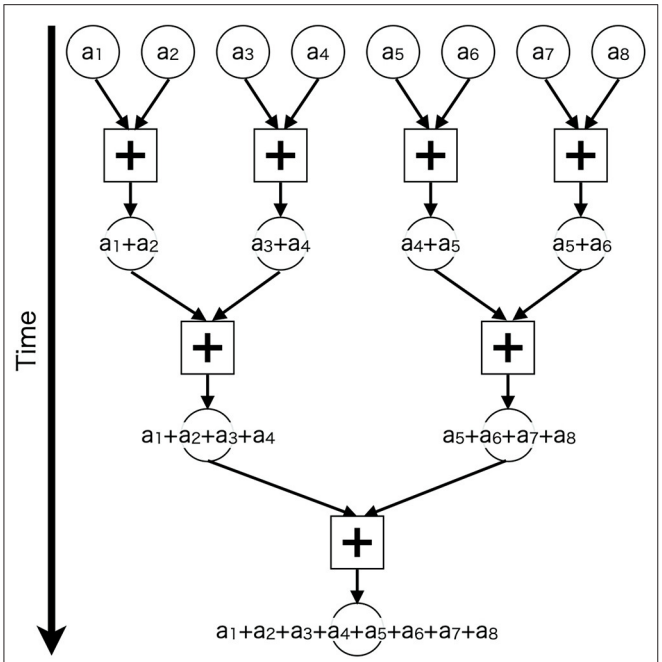

FIGURE 3 | Schematic of parallel reduction. Circles represent values to be added (a_1–a_8) and squares represent add operations (+) assigned to threads. Each thread adds two values in parallel, and the additions are repeated recursively.

reflex by which the eyes rotate to the same direction with the visual world's movement so as to reduce the image slip on the retina. If eye movement is too small against the visual movement, an image slip occurs on the retina. The slip generates an error signal fed to PCs via CFs, which in turn induces long-term depression (LTD) at PF-PC synapses. The LTD leads to larger eye movement gain, and further image slips are prevented. This is called gain adaptation of OKR.

To simulate the gain adaptation OKR, we added an inferior olive (IO) to the cerebellar scaffold model and bidirectional plasticity (LTD and long-term potentiation, LTP) at PF-PC synapses as follows:

$$w_{\mathrm{PC}_i,\mathrm{PF}_j}(t+1) = w_{\mathrm{PC}_i,\mathrm{PF}_j}(t) + 0.001\left(w_{\mathrm{init}} - w_{\mathrm{PC}_i,\mathrm{PF}_j}(t)\right) - 0.01 w_{\mathrm{PC}_i,\mathrm{PF}_j}(t)\sum_{\Delta t=0}^{50}\mathrm{CF}(t)\mathrm{PF}_j(t-\Delta t) \quad (3)$$

where $\mathrm{PF}_j(t)$ and $\mathrm{CF}(t)$ take 1 if PF_j or CF elicited a spike at time t, and 0 otherwise. The 2nd term on the RHS simulates LTP that occurred by firing of a presynaptic PF only (Sakurai, 1987). The 3rd term simulates LTD by conjunctive activation of a CF and a PF (Ito, 2001), that are active 0–50 ms earlier than the CF activation. The w_{init} is a constant value representing the initial synaptic weight and was set at 1.0. A synaptic weight for each pair of a PF and a PC is assigned to a thread and updated by the thread independently with every 1 ms.

In the simulation, we fed Poisson spikes that the firing rate modulates from 0 to 30 Hz sinusoidally in 2 s to MFs and that from 0 to 3 Hz to the IO to simulate the visual world movements and the retinal slip errors according to electrophysiological findings (Nagao, 1988). We repeatedly fed these spikes 300 times, and recorded the changes of the firing rates of PCs and DCN cells. Parameters were adjusted according to our previous OKR adaptation study (Yamazaki and Nagao, 2012), since the parameters were different from Casali et al. (2019). We did not include nucleo-olivery inhibitory connections.

3. RESULTS

3.1. Reimplementation of the Cerebellar Scaffold Model

We have successfully reimplemented the cerebellar scaffold model. Specifically, we implemented the simulation module of the scaffold model on GPUs, whereas the other modules were left unchanged.

First, we compared the basic dynamics of the previous and present models by presenting the same MF stimuli. The stimulation setting was identical to that in the original paper (Casali et al., 2019) described in section 2.4. Specifically, we plotted spike activities for GrC, GoC, BC, SC, PC, and DCN of the previous (Casali et al., 2019) and present versions, respectively, in response to the MF stimuli (**Figure 4A**). These raster plots look similar. Then, to quantify the similarity, we calculated the mean firing rates for each neuron type and their standard deviations (**Figure 4B**). We found that the mean firing rates in one model fall within the range of the standard deviation of the other Here, it should be emphasized that the scaffolding approach intends not to reproduce statistically non-significant results. We will discuss this issue further in section 4. These results suggest that the present model was able to reproduce the network activity of the previous model.

3.2. Computational Time

Next, we compared computational speeds of the previous and present models. To do so, we fed the same MF stimuli as in the previous section, followed by an additional 9 s of the background noise at 1 Hz, and measured the computational time for the simulation (**Figure 5**). The previous study reported that a simulation spent 570 s on a single computer node of a supercomputer, which was accelerated to 278 s on 4 nodes on the same computer. On the other hand, our model on a single GPU spent only 2.62 ± 0.15s, resulting in ∼217 and 106 times faster speeds, respectively. These results indicate that our model runs four times faster than real time.

3.3. Weak and Strong Scaling Properties

We then examined weak and strong scaling properties up to 4 GPUs. In both weak and strong scaling, we obtained good scaling properties (**Figures 6A,B**), respectively. However, the good scaling properties depend on how we split the network into multiple smaller networks to be assigned to different GPUs. We will discuss this issue in section 4.

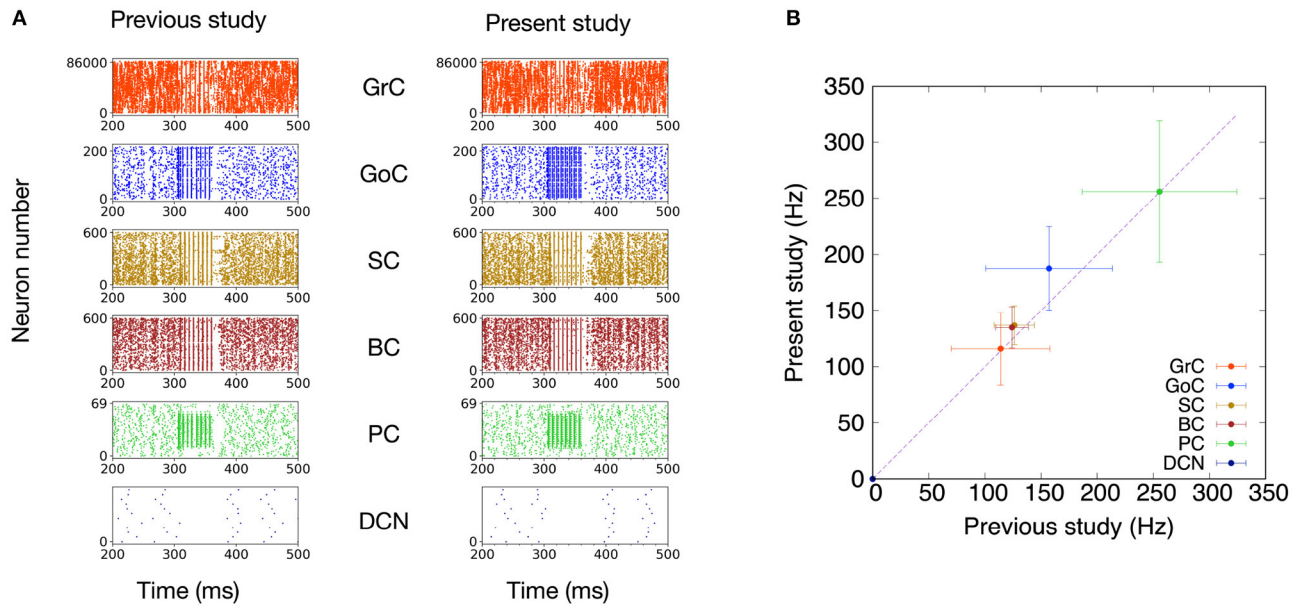

FIGURE 4 | Comparison of simulated neuronal activity between the PyNEST version of the previous study (Casali et al., 2019) and the present study. **(A)** Raster plots of neuron populations in response to MF stimuli (50 ms at 150 Hz on 2,932 MFs) superimposed on 1-Hz background noise. **(B)** Comparison of mean firing rates (Hz) during the input stimulation. The plots compare mean firing rate during the input stimulation for the different neuron populations of the present study with those reported by the previous study (Casali et al., 2019). Dashed line shows identity. Error bars show the standard deviations.

3.4. Faster-Than-Real-Time Simulation of OKR Gain Adaptation

Finally, we conducted computer simulation of OKR gain adaptation. To simulate synaptic plasticity at PF-PC synapses, we added an IO cell. In response to sinusoidal MF signal, PCs exhibited a sinusoidal activity pattern with the opposite phase, whereas DCNs were activated in phase. Furthermore, during 300 trials, PCs decreased the firing rate, where the minimal firing rate changed from 60 to 24 Hz (**Figure 7A**). On the other hand, DCNs increased the firing rate from 105 to 124 Hz (**Figure 7B**), which could correspond to gain increase in OKR (Nagao, 1988). Furthermore, a simulation of a 2 s trial was completed within 750 ms, suggesting that faster-than-real-time simulation was achieved.

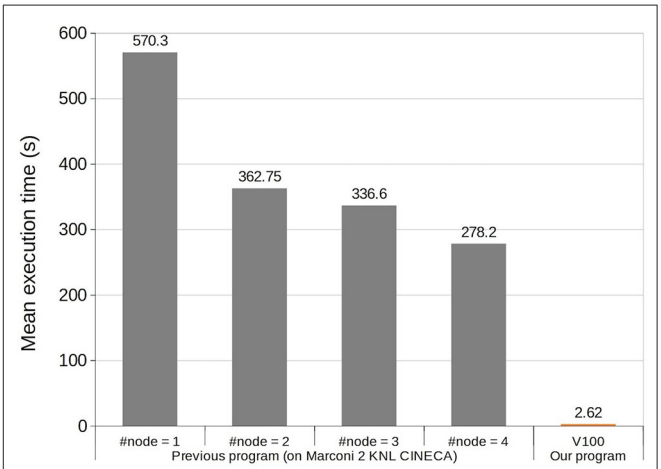

FIGURE 5 | Execution time comparison of 10 s simulation between the previous model (Casali et al., 2019) and ours. The horizontal axis represents four different settings of computers with 1–4 nodes used in the previous study, and our V100 GPU. The vertical axis represents mean simulation time.

4. DISCUSSION

4.1. Proof of Concept of the Scaffolding Approach

The scaffolding approach aims to make neuron models, placements, connectivity, and numerical methods replaceable without affecting the others. Our study is a proof of concept of this approach. In fact, we were able to replace the original simulation module with our version written from scratch that worked efficiently on GPUs. We were able to accelerate simulations faster than real time, while reproducing the same simulation results with the previous study, and further obtain good scaling. Real-time simulation is considered a milestone for simulation speed. Faster than real-time simulation will allow further elaboration of the present model while still keeping real time simulation. These results suggest that the scaffolding approach is useful as a compromise between flexibility of modeling and efficiency of simulation.

4.2. Better Decomposition of a Network for Better Scaling

To obtain good parallel computing performance and scaling, it is crucial to consider how to decompose a large network into

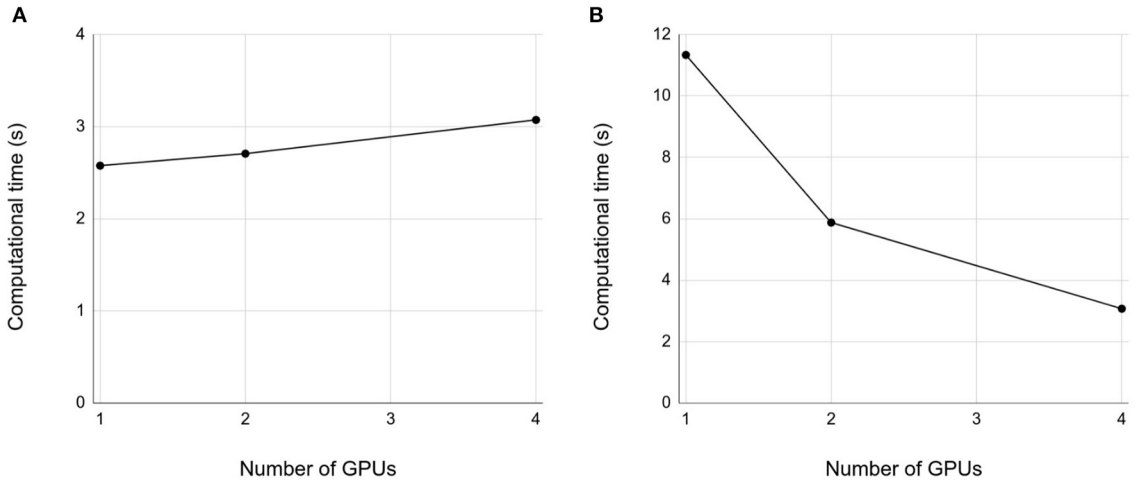

FIGURE 6 | Weak and strong properties. **(A)** Weak scaling. Each GPU simulates the 10 s spontaneous activity of the cerebellar network in 200 × 800 × 900μm^3 volume. The horizontal axis is the number of GPUs, and vertical axis is the computational time. **(B)** Strong scaling. A network of the size 800 × 800 μm^3 volume was decomposed into 1, 2, and 4 subnetworks for 1, 2, and 4 GPUs, respectively. Conventions as in **(A)**.

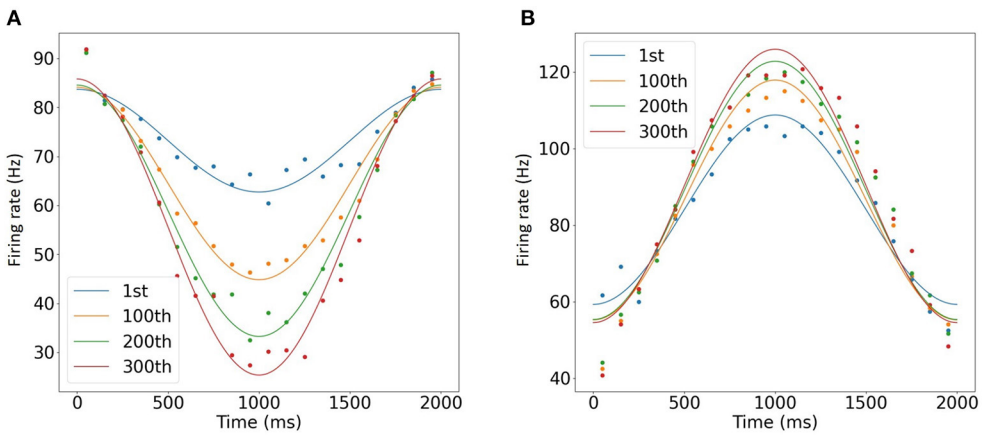

FIGURE 7 | Simulation of OKR adaptation. **(A,B)** Learned change of the mean firing rates of PCs **(A)** and DCNs **(B)** at the 1st, 100th, 200th, and 300th trials of MF signal oscillation (blue, orange, green, and purple, respectively). Dots represent firing rates every 100ms, whereas lines show the data fitted with cosine functions. The horizontal axis represents time for 1 trial, and vertical axis represents firing rate in Hz.

a set of smaller subnetworks. In neural network simulation, three types of decomposition have been used: random, 2-D, and 1-D. In random decomposition, a network is split into multiple subnetworks that contain almost the same numbers of neurons and synapses for load balancing. This method is usable for any type of neural network, even a network that does not have spatial structures. The NEST simulator has been using this method (Jordan et al., 2018). In 2-D decomposition, a network with spatial cellular and synaptic placements is split into multiple smaller 2-D tiles. This method is useful when a network has spatially regular connections spanning in 2-D or 3-D, and the connections are rather short. The MONET simulator uses this method (Igarashi et al., 2019; Yamaura et al., 2020). In this study, we used 1-D decomposition. We split our network model along the transversal axis. This was efficient for cerebellar networks, because in the cerebellar cortex, PFs run several mm along transversal direction in parallel (Eccles et al., 1967), suggesting that splitting the network in the perpendicular direction (i.e., sagittal direction) result in a large number of communications for spike propagation. However, when we are able to simulate a sufficient size of a network that is larger than a few cm on a GPU, 2-D decomposition will be better than 1-D, because PF spikes do not propagate longer than 1 cm.

4.3. Comparison With Other Models

Various computational models of the cerebellar microcircuit have been proposed. Medina et al. (2000) was the first large-scale spiking network model with 10,000 neurons for delay eyeblink conditioning, which was later reimplemented on a single GPU with 1 million neurons (Li et al., 2013). Yamazaki and

Tanaka (2007) and Yamazaki and Nagao (2012) built another spiking network model with 100,000 neurons that integrates gain and timing learning mechanisms. These models were also reimplemented on a single GPU (Yamazaki and Igarashi, 2013), multi GPUs (Gosui and Yamazaki, 2016), and a supercomputer with 1,024 PEZY-SC processors (Yamazaki et al., 2019).

All these GPU versions ran in real time as did the present study. The last one was composed of 0.1 billion neurons while retaining real-time simulation. That is roughly the same size as a cat's entire cerebellum. The simulation codes of these models were written by the authors specially from scratch without using general-purpose simulators. Writing special codes enables us to harness the maximal performance of computers while giving up flexibility and reusability of the codes. On the other hand, by using general-purpose simulators, scientists can focus on modeling while ignoring all the other issues which are not related to neuroscience, such as writing programs, debugging, numerical methods, accuracy, and stability. Yamaura et al. (2020) used a general-purpose simulator MONET and built a human-scale network model composed of 68 billion spiking neurons on the K computer, which was the previous Japanese flagship supercomputer, although the computer simulation was about 600 times slower than real time. A drawback of using general-purpose simulators is much lower performance compared with special-purpose programs.

The scaffolding approach, which was originally proposed by Casali et al. (2019) and augmented in the present study, is an intermediate approach. In this approach, simulation codes are decomposed into a few functional modules which are almost independent but loosely connected. Each module can be either written from scratch as in the present study or implemented using existing simulators as in Casali et al. (2019). In fact, our simulation module is reusable for other types of networks that are generated by a connectivity module. A similar approach has been demonstrated by Brian2GeNN (Stimberg et al., 2020), which generates a custom C++ code with GPU support from a script written for Brian simulator (Goodman and Brette, 2008). While the scaffolding approach separates cell placement and connectivity into different modules, Brian integrates them. In this sense, the scaffolding approach might be more flexible. Meanwhile, Brian2GeNN supports only a single GPU at the time of writing. In this way, the scaffolding approach provides a good compromise between flexibility and performance.

Furthermore, the scaffolding approach may allow us to elaborate a model gradually as easily as general simulators, while realizing efficient simulation comparable to custom-code simulators. For example, single-compartment models with more realistic internal parameters have already been integrated in a scaffold model (Geminiani et al., 2019). Moreover, multi-compartment neuron models, such as a PC model (De Schutter and Bower, 1994a,b; Masoli et al., 2015; Masoli and D'Angelo, 2017), a GoC model (Solinas et al., 2007a,b), a GrC model (Diwakar et al., 2009; Dover et al., 2016; Masoli et al., 2020), and IO models (Schweighofer et al., 1999; De Gruijl et al., 2012), will be integrated. Integrating these elaborated models would allow us investigate more detailed network dynamics including synaptic plasticity (Casali et al., 2020) as well as intracellular dynamics simultaneously.

4.4. Limitations

Several limitations exist on our simulation module and the scaffolding approach itself.

Our simulation module so far implements only conductance-based LIF model for neurons, conductance-based exponential decay synapses, and Euler methods for numerical integration so as to simulate the cerebellar model implemented in Casali et al. (2019). Such simple implementation helped to achieve real-time simulation. On the contrary, if we implement nonlinear models such as Hodgkin-Huxley type models and more precise numerical methods such as a 4th-order Runge-Kutta method, they may slow down numerical simulation than real time. However, currently 95% of execution time is occupied by product-sum operations for synaptic inputs. Therefore, incorporating more elaborated neuron/synapse models and precise numerical methods will not affect the simulation time significantly. Furthermore, our simulation module includes some model-specific optimizations for the cerebellar model such as parallel reduction, 1-D decomposition, and DCNs' simulation at host. Applying these optimizations to other types of network models would cause a slow down.

On the other hand, when a network model employs randomness in the simulation such as input noise, the scaffolding approach does not guarantee individual spike timing-level reproducibility across different simulation modules, because randomness is managed by individual simulation modules. The scaffolding approach intends not to provide completely identical results across multiple simulation modules, but to only provide statistically non-significant results.

4.5. Future Directions

We are now able to simulate in real time a cerebellar circuit with multiple functional modules, at as large a scale as we have a sufficient number of GPUs, owing to the weak-scaling property. The model will be able to run efficiently on supercomputers with multiple GPUs, such as JUWELS (Forschungszentrum Jülich, 2019) and JURECA (Krause and Thörnig, 2016) in Julich supercomputing center, and Germany managed under the Human Brain Project.

Because the model will consist of multiple functional modules, which could learn internal models independently, the model will provide a means to investigate how multiple internal models work together synergistically to perform a complex task, which has been postulated by MOSAIC models (Wolpert and Kawato, 1998; Haruno et al., 2001). Contrary to the modular computing approach, Michikawa et al. (2020) recently reported that an ultra wide-field Ca^{2+} imaging procedure over the entire cerebellar cortex revealed that all microzones are always activated simultaneously, suggesting that all cerebellar modules function in a holistic manner. This study implies that many rather than a small subset of microcomplexes share functions for a given task, but how? To address these questions computationally, our cerebellar model would be a useful tool. Moreover, we could even replace neuron models with more elaborated versions, which would be important for studying how intracellular Ca^{2+} signals spread over the entire cerebellar cortex.

Furthermore, real-time computing capability will also allow us to adopt the cerebellar model as a controller of hardware

robots, as in the previous studies (Garrido et al., 2013; Yamazaki and Igarashi, 2013; Naveros et al., 2014; Casellato et al., 2015; Pinzon-Morales and Hirata, 2015; Xu et al., 2018). An advantage of the present model over the previous models is that although the previous models could control only one parameter, because they have only one output channel, the present model can control multiple parameters. This will enable us to control complex robots that have many degrees of freedom. Furthermore, learning capability of the present model would enable such robots to learn appropriate behaviors autonomously. Although traditionally the cerebellum is considered a supervised learning machine that requires an external teacher, a recent study proposes that the cerebellum is a reinforcement learning machine (Yamazaki and Lennon, 2019) that does not require such teachers. The cerebellum as a reinforcement learning machine will be a promising candidate for such autonomous robot learning. Along with the development of spiking neural networks acting as learning machines, development of hardware for emulating such spiking neural networks has been advanced rapidly (Monroe, 2014). Such hardware called "neuromorphic processors" aim to emulate spiking neural networks in real time with ultra low power for mainly edge computing (Furber et al., 2014; Merolla et al., 2014; Friedmann et al., 2017; Davies et al., 2018; DeBole et al., 2019). Machine learning study and computational neuroscience study would be integrated more tightly for real world applications.

5. CONCLUSION

The present cerebellar model is flexible and extendable, owing to the scaffolding approach. It is also efficient, owing to the parallel computing capability of GPUs. The model will provide a means to further investigate the role of the cerebellum in motor or non-motor learning computationally. Finally, the source code is available at the author's GitHub[1].

[1]https://github.com/Rkuriyama/Cerebellar-Scaffold-Model-Simulation-on-GPU/

DATA AVAILABILITY STATEMENT

The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation.

AUTHOR CONTRIBUTIONS

TY and ED'A conceived and designed the research. CC contributed the source code of the original scaffold model. RK wrote the code, performed simulation, and analyzed the data. RK and TY wrote the original draft. RK, CC, ED'A, and TY discussed on the draft and revised for the final version. All authors contributed to the article and approved the submitted version.

FUNDING

This work was supported by MEXT/JSPS KAKENHI Grant numbers JP17H06310, JP17K07049, and JP20K06850, and MEXT Post-K Exploratory Challenge #4-1 (hp190146). Part of this study was based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO). This project has received funding from the European Union's Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreement No. 785907 (Human Brain Project SGA2), and the Specific Grant Agreement No. 945539 (Human Brain Project SGA3).

ACKNOWLEDGMENTS

We thank Dr. Hiroshi Yamaura, Mr. Taira Kobayashi, and Ryohei Hoashi for fruitful discussions.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fncel.2021.623552/full#supplementary-material

REFERENCES

Amunts, K., Ebell, C., Muller, J., Telefont, M., Knoll, A., and Lippert, T. (2016). The human brain project: creating a European research infrastructure to decode the human brain. *Neuron* 92, 574–581. doi: 10.1016/j.neuron.2016.10.046

Brette, R., Lilith, M., Carnevale, T., Hines, M., Beeman, D., Bower, J., et al. (2008). Simulation of networks of spiking neurons: a review of tools and strategies. *J. Comput. Neurosci.* 23, 349–398. doi: 10.1007/s10827-007-0038-6

Carnevale, N. T., and Hines, M. L. (2006). *The NEURON Book*. Cambridge: Cambridge University Press. doi: 10.1017/CBO9780511541612

Casali, S., Marenzi, E., Medini, C., Casellato, C., and D'Angelo, E. (2019). Reconstruction and simulation of a scaffold model of the cerebellar network. *Front. Neuroinform.* 13:37. doi: 10.3389/fninf.2019.00051

Casali, S., Tognolina, M., Gandolfi, D., Mapelli, J., D'Angelo, E. (2020). Cellular-resolution mapping uncovers spatial adaptive filtering at the rat cerebellum input stage. *Commun. Biol.* 3:635. doi: 10.1038/s42003-020-01360-y

Casellato, C., Antonietti, A., Garrido, J., Ferrigno, G., D'Angelo, E., and Pedrocchi, A. (2015). Distributed cerebellar plasticity implements generalized multiple-scale memory components in real-robot sensorimotor tasks. *Front. Comput. Neurosci.* 9:24. doi: 10.3389/fncom.2015.00024

Davies, M., Srinivasa, N., Lin, T. H., Chinya, G., Cao, Y., Choday, S. H., et al. (2018). Loihi: a neuromorphic manycore processor with on-chip learning. *IEEE Micro* 38, 82–99. doi: 10.1109/MM.2018.112130359

De Gruijl, J. R., Bazzigaluppi, P., de Jeu, M. T. G., and De Zeeuw, C. I. (2012). Climbing fiber burst size and olivary sub-threshold oscillations in a network setting. *PLoS Comput. Biol.* 8:e1002814. doi: 10.1371/journal.pcbi.1002814

De Schutter, E., and Bower, J. M. (1994a). An active membrane model of the cerebellar Purkinje cell I. simulation of current clamps in slice. *J. Neurophysiol.* 71, 375–400. doi: 10.1152/jn.1994.71.1.375

De Schutter, E., and Bower, J. M. (1994b). An active membrane model of the cerebellar Purkinje cell II. simulation of synaptic responses. *J. Neurophysiol.* 71, 401–419. doi: 10.1152/jn.1994.71.1.401

DeBole, M. V., Taba, B., Amir, A., Akopyan, F., Andreopoulos, A., Risk, W. P., et al. (2019). Truenorth: accelerating from zero to 64 million neurons in 10 years. *Computer* 52, 20–29. doi: 10.1109/MC.2019.2903009

Diwakar, S., Magistretti, J., Goldfarb, M., Naldi, G., and D'Angelo, E. (2009). Axonal Na+ channels ensure fast spike activation and back-propagation in cerebellar granule cells. *J. Neurophysiol.* 101, 519–532. doi: 10.1152/jn.90382.2008

Dover, K., Marra, C., Solinas, S., Popovic, M., Subramaniyam, S., Zecevic, D., et al. (2016). FHF-independent conduction of action potentials along the leak-resistant cerebellar granule cell axon. *Nat. Commun.* 7:12895. doi: 10.1038/ncomms12895

Eccles, J. C., Ito, M., and Szentágothai, J. (1967). *The Cerebellum as a Neuronal Machine*. New York, NY: Springer. doi: 10.1007/978-3-662-13147-3

Eppler, J., Helias, M., Muller, E., Diesmann, M., and Gewaltig, M.-O. (2009). Pynest: a convenient interface to the nest simulator. *Front. Neuroinform.* 2:12. doi: 10.3389/neuro.11.012.2008

Forschungszentrum Jülich (2019). JUWELS: modular tier-0/1 supercomputer at jülich supercomputing centre. *J. Large Scale Res. Facilit.* 5:A135. doi: 10.17815/jlsrf-5-171

Friedmann, S., Schemmel, J., Grübl, A., Hartel, A., Hock, M., and Meier, K. (2017). Demonstrating hybrid learning in a flexible neuromorphic hardware system. *IEEE Trans. Biomed. Circ. Syst.* 11, 128–142. doi: 10.1109/TBCAS.2016.2579164

Furber, S. B., Galluppi, F., Temple, S., and Plana, L. A. (2014). The spinnaker project. *Proc. IEEE* 102, 652–665. doi: 10.1109/JPROC.2014.2304638

Garrido, J. A., Luque, N., D'Angelo, E., and Ros, E. (2013). Distributed cerebellar plasticity implements adaptable gain control in a manipulation task: a closed-loop robotic simulation. *Front. Neural Circuits* 7:159. doi: 10.3389/fncir.2013.00159

Geminiani, A., Pedrocchi, A., D'Angelo, E., and Casellato, C. (2019). Response dynamics in an olivocerebellar spiking neural network with non-linear neuron properties. *Front. Comput. Neurosci.* 13:68. doi: 10.3389/fncom.2019.00068

Gewaltig, M.-O., and Diesmann, M. (2007). NEST (NEural Simulation Tool). *Scholarpedia* 2:1430. doi: 10.4249/scholarpedia.1430

Goodman, D., and Brette, R. (2008). Brian: a simulator for spiking neural networks in python. *Front. Neuroinform.* 2:5. doi: 10.3389/neuro.11.005.2008

Gosui, M., and Yamazaki, T. (2016). Real-world-time simulation of memory consolidation in a large-scale cerebellar model. *Front. Neuroanat.* 10:21. doi: 10.3389/fnana.2016.00021

Harris, M. (2011). *Optimizing Parallel Reduction in CUDA*. Available online at: https://developer.download.nvidia.com/assets/cuda/files/reduction.pdf (accessed March 14, 2020).

Haruno, M., Wolpert, D. M., and Kawato, M. (2001). Mosaic model for sensorimotor learning and control. *Neural Comput.* 13, 2201–2220. doi: 10.1162/089976601750541778

Hines, M., Kumar, S., and Schürmann, F. (2011). Comparison of neuronal spike exchange methods on a Blue Gene/P supercomputer. *Front. Comput. Neurosci.* 5:49. doi: 10.3389/fncom.2011.00049

Igarashi, J., Yamaura, H., and Yamazaki, T. (2019). Large-scale simulation of a layered cortical sheet of spiking network model using a tile partitioning method. *Front. Neuroinform.* 13:71. doi: 10.3389/fninf.2019.00071

Ito, M. (1984). *The Cerebellum and Neural Control*. New York, NY: Raven Press.

Ito, M. (2001). Cerebellar long-term depression: characterization, signal transduction, and functional roles. *Physiol. Rev.* 81, 1143–1195. doi: 10.1152/physrev.2001.81.3.1143

Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Trans. Neural Netw.* 14, 1569–1572. doi: 10.1109/TNN.2003.820440

Jordan, J., Ippen, T., Helias, M., Kitayama, I., Sato, M., Igarashi, J., et al. (2018). Extremely scalable spiking neuronal network simulation code: from laptops to exascale computers. *Front. Neuroinform.* 12:2. doi: 10.3389/fninf.2018.00034

Krause, D., and Thörnig, P. (2016). JURECA: General-purpose supercomputer at julich supercomputing centre. *J. Large-Scale Res. Facil.* 2:A62. doi: 10.17815/jlsrf-2-121

Kumbhar, P., Hines, M., Fouriaux, J., Ovcharenko, A., King, J., Delalondre, F., et al. (2019). CoreNEURON : an optimized compute engine for the NEURON simulator. *Front. Neuroinform.* 13:63. doi: 10.3389/fninf.2019.00063

Li, W., Hausknecht, M., Stone, P., and Mauk, M. (2013). Using a million cell simulation of the cerebellum: network scaling and task generality. *Neural Netw.* 47, 95–102. doi: 10.1016/j.neunet.2012.11.005

Manssen, M., Weigel, M., and Hartmann, A. K. (2012). Random number generators for massively parallel simulations on GPU. *Eur. Phys. J. Spec. Top.* 210, 53–71. doi: 10.1140/epjst/e2012-01637-8

Masoli, S., and D'Angelo, E. (2017). Synaptic activation of a detailed purkinje cell model predicts voltage-dependent control of burst-pause responses in active dendrites. *Front. Cell. Neurosci.* 11:278. doi: 10.3389/fncel.2017.00278

Masoli, S., Solinas, S., and D'Angelo, E. (2015). Action potential processing in a detailed Purkinje cell model reveals a critical role for axonal compartmentalization. *Front. Cell. Neurosci.* 9:47. doi: 10.3389/fncel.2015.00047

Masoli, S., Tognolina, M., Laforenza, U., Moccia, F., and D'Angelo, E. (2020). Parameter tuning differentiates granule cell subtypes enriching transmission properties at the cerebellum input stage. *Commun. Biol.* 3:222. doi: 10.1038/s42003-020-0953-x

Medina, J. F., Garcia, K. S., Nores, W. L., Taylor, N. M., and Mauk, M. D. (2000). Timing mechanisms in the cerebellum: testing predictions of a large-scale computer simulation. *J. Neurosci.* 20, 5516–5525. doi: 10.1523/JNEUROSCI.20-14-05516.2000

Merolla, P. A., Arthur, J. V., Alvarez-Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., et al. (2014). A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* 345, 668–673. doi: 10.1126/science.1254642

Michikawa, T., Yoshida, T., Kuroki, S., Ishikawa, T., Kakei, S., Itohara, S., et al. (2020). Distributed sensory coding by cerebellar complex spikes in units of cortical segments. *bioRxiv [Preprint]*. doi: 10.1101/2020.09.18.301564

Monroe, D. (2014). Neuromorphic computing gets ready for the (really) big time. *Commun. ACM* 57, 13–15. doi: 10.1145/2601069

Nagao, S. (1988). Behavior of floccular purkinje cells correlated with adaptation of horizontal optokinetic eye movement response in pigmented rabbits. *Exp. Brain Res.* 73, 489–497. doi: 10.1007/BF00406606

Naveros, F., Luque, N. R., Garrido, J., Carrillo, R. R., Anguita, M., and Ros, E. (2014). A spiking neural simulator integrating event-driven and time-driven computation schemes using parallel CPU-GPU co-processing: a case study. *IEEE Trans. Neural Netw. Learn. Syst.* 26, 1567–1574. doi: 10.1109/TNNLS.2014.2345844

NVIDIA (2017). *NVIDIA Tesla v100 GPU Architecture*. Available online at: https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf (accessed March 19, 2020).

NVIDIA (2020a). *CUDA Toolkit Documentation*. Available online at: https://docs.nvidia.com/cuda/ (accessed February 15, 2020).

NVIDIA (2020b). *NVIDIA DGX Station Datasheet*. Available online at: https://www.nvidia.com/en-us/data-center/dgx-station/ (accessed March 19, 2020).

Pinzon-Morales, R.-D., and Hirata, Y. (2015). A realistic bi-hemispheric model of the cerebellum uncovers the purpose of the abundant granule cells during motor control. *Front. Neural Circ.* 9:18. doi: 10.3389/fncir.2015.00018

Ramos, K. M., Grady, C., Greely, H. T., Chiong, W., Eberwine, J., Farahany, N. A., et al. (2019). The NIH BRAIN initiative: Integrating neuroethics and neuroscience. *Neuron* 101, 394–398. doi: 10.1016/j.neuron.2019.01.024

Sakurai, M. (1987). Synaptic modification of parallel fibre-purkinje cell transmission in *in vitro* guinea-pig cerebellar slices. *J. Physiol.* 394, 463–480. doi: 10.1113/jphysiol.1987.sp016881

Schweighofer, N., Doya, K., and Kawato, M. (1999). Electrophysiological properties of inferior olive neurons: a compartmental model. *J. Neurophysiol.* 82, 804–817. doi: 10.1152/jn.1999.82.2.804

Shahnaz, R., Usman, A., and Chughtai, I. R. (2005). "Review of storage techniques for sparse matrices," in *2005 Pakistan Section Multitopic Conference*, 1–7. doi: 10.1109/INMIC.2005.334453

Solinas, S., Forti, L., Cesana, E., Mapelli, J., De Schutter, E., and D'Angelo, E. (2007a). Computational reconstruction of pacemaking and intrinsic electroresponsiveness in cerebellar golgi cells. *Front. Cell. Neurosci.* 1:2. doi: 10.3389/neuro.03.002.2007

Solinas, S., Forti, L., Cesana, E., Mapelli, J., De Schutter, E., and D'Angelo, E. (2007b). Fast-reset of pacemaking and theta-frequency resonance patterns in cerebellar golgi cells: simulations of their impact *in vivo*. *Front. Cell. Neurosci.* 1:4. doi: 10.3389/neuro.03.004.2007

Stimberg, M., Goodman, D. F. M., and Nowotny, T. (2020). Brian2GeNN: accelerating spiking neural network simulations with graphics hardware. *Sci. Rep.* 10:410. doi: 10.1038/s41598-019-54957-7

Wolpert, D. M., and Kawato, M. (1998). Multiple paired forward and inverse models for motor control. *Neural Netw*. 11, 1317–1329. doi: 10.1016/S0893-6080(98)00066-5

Xu, T., Xiao, N., Zhai, X., Chan, P. K., and Tin, C. (2018). Real-time cerebellar neuroprosthetic system based on a spiking neural network model of motor learning. *J. Neural Eng*. 15:016021. doi: 10.1088/1741-2552/aa98e9

Yamaura, H., Igarashi, J., and Yamazaki, T. (2020). Simulation of a human-scale cerebellar network model on the k computer. *Front. Neuroinform*. 14:16. doi: 10.3389/fninf.2020.00016

Yamazaki, T., and Igarashi, J. (2013). Realtime cerebellum: a large-scale spiking network model of the cerebellum that runs in realtime using a graphics processing unit. *Neural Netw*. 47, 103–111. doi: 10.1016/j.neunet.2013.01.019

Yamazaki, T., Igarashi, J., Makino, J., and Ebisuzaki, T. (2019). Real-time simulation of a cat-scale artificial cerebellum on PEZY-SC processors. *Int. J. High Perf. Comp. App*. 33, 155–168. doi: 10.1177/1094342017710705

Yamazaki, T., Igarashi, J., and Yamaura, H. (in press). Human-scale brain simulation via supercomputer: a case study on the cerebellum. *Neuroscience*. doi: 10.1016/j.neuroscience.2021.01.014

Yamazaki, T., and Lennon, W. (2019). Revisiting a theory of cerebellar cortex. *Neurosci. Res*. 148, 1–8. doi: 10.1016/j.neures.2019.03.001

Yamazaki, T., and Nagao, S. (2012). A computational mechanism for unified gain and timing control in the cerebellum. *PLoS ONE* 7:e33319. doi: 10.1371/journal.pone.0033319

Yamazaki, T., and Tanaka, S. (2007). A spiking network model for passage-of-time representation in the cerebellum. *Eur. J. Neurosci*. 26, 2279–2292. doi: 10.1111/j.1460-9568.2007.05837.x

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.