Neuroinformatics (2020) 18:407–428
https://doi.org/10.1007/s12021-019-09451-w

ORIGINAL ARTICLE

Understanding Computational Costs of Cellular-Level Brain Tissue Simulations Through Analytical Performance Models

Francesco Cremonesi[1] · Felix Schürmann[1]

Published online: 13 February 2020

Abstract

Computational modeling and simulation have become essential tools in the quest to better understand the brain's makeup and to decipher the causal interrelations of its components. The breadth of biochemical and biophysical processes and structures in the brain has led to the development of a large variety of model abstractions and specialized tools, often times requiring high performance computing resources for their timely execution. What has been missing so far was an in-depth analysis of the complexity of the computational kernels, hindering a systematic approach to identifying bottlenecks of algorithms and hardware. If whole brain models are to be achieved on emerging computer generations, models and simulation engines will have to be carefully co-designed for the intrinsic hardware tradeoffs. For the first time, we present a systematic exploration based on analytic performance modeling. We base our analysis on three *in silico* models, chosen as representative examples of the most widely employed modeling abstractions: current-based point neurons, conductance-based point neurons and conductance-based detailed neurons. We identify that the synaptic modeling formalism, i.e. current or conductance-based representation, and not the level of morphological detail, is the most significant factor in determining the properties of memory bandwidth saturation and shared-memory scaling of *in silico* models. Even though general purpose computing has, until now, largely been able to deliver high performance, we find that for all types of abstractions, network latency and memory bandwidth will become severe bottlenecks as the number of neurons to be simulated grows. By adapting and extending a performance modeling approach, we deliver a first characterization of the performance landscape of brain tissue simulations, allowing us to pinpoint current bottlenecks for state-of-the-art *in silico* models, and make projections for future hardware and software requirements.

Keywords Computational models of neurons · Brain tissue simulations · Performance modeling · High performance computing

Introduction

In the field of computational neuroscience, simulations of biological neural networks represent one of the fundamental tools for hypothesis testing and exploration. A widely used scale of representation are neuron-based approaches, i.e. models of brain tissue in which the fundamental unit is represented by a neuronal cell. This representation is important as it allows for a faithful matching of the model with a range of anatomical and electrophysiological data (Markram et al. 2015; Potjans and Diesmann 2012; Pozzorini et al. 2015; Hagen et al. 2016, 2018). While determining the adequate level of detail is a formidable challenge with respect to the system modeled, so is the addressing of an efficient implementation of the simulation in software. The size of such networks, both in terms of number of neurons and synapses and rate of synaptic events, as well as the level of biological detail required to answer meaningful questions about the brain, mean that these simulations come at a large computational cost.

Much of the early increase in computational requirements of models and simulations have been supported by

This study was supported by funding to the Blue Brain Project, a research center of the École polytechnique fédérale de Lausanne (EPFL), from the Swiss governments ETH Board of the Swiss Federal Institutes of Technology

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s12021-019-09451-w) contains supplementary material, which is available to authorized users.

✉ Felix Schürmann
felix.schuermann@epfl.ch

[1] Blue Brain Project, Brain Mind Institute, École polytechnique fédérale de Lausanne (EPFL), Campus Biotech, 1202 Geneva, Switzerland

Dennard scaling (1974) and Moore's law (1995), but with chip-manufacturing technology reaching its limit and the consequent rise of multi-core and heterogeneous architectures (Hardavellas et al. 2011; Simonite 2016), computational neuroscientists have been forced to develop more efficient algorithms and software to be able to keep up with the increasing demands of modellers. Research efforts in the context of simulation neuroscience have investigated the efficient utilization of modern multicore processors (see e.g. Kumbhar et al. 2016, 2018; Eichner et al. 2009; Brette and Goodman 2011), parallel computing (see e.g. Morrison et al. 2005; Ovcharenko et al. 2015; Helias et al. 2012), accelerators (see e.g. Knight and Nowotny 2018; Fidjeland et al. 2009; Brette and Goodman 2012) and brain-inspired hardware (see e.g. Painkras et al. 2013; Benjamin et al. 2014; Indiveri et al. 2011).

Despite the multiple years of research in efficient implementations of neuron models, we are still missing a more quantitative treatment of what are the actual computational characteristics of a given level of detail and how a particular level of detail may be limited by specific hardware trade-offs. How much more costly is a morphologically detailed neuron simulation compared to a representation modeling the same neuron as a point? What is the influence of how the synapses are being modeled? Can we expect that point neuron models can scale to massively parallel computers in a similar way than detailed neuron models?

For the first time, we extend performance modeling techniques to the field of computational neuroscience, allowing us to establish a quantitative relationship between the parameters dictated by the biophysical model, the complexity properties of the simulation algorithm and the details of the hardware specifications. Although we require a reasonable level of accuracy and validation against benchmarks, our goal is not to obtain highly accurate performance predictions, but rather to design a tool with sufficient generality to identify current and future bottlenecks for different levels of abstraction on the spectrum of models for neural cells' dynamics. Based on our requirements, we choose a performance modeling approach that is neither purely based on first-principles (see e.g. Williams et al. 2009) nor purely empirical (see e.g. Calotoiu et al. 2013) but is instead an hybrid approach known as grey-box analytical modeling. Our analysis is based on state-of-the-art high performance computing (HPC) hardware architecture and applied to three published neural network simulations that have been selected to represent the diversity of neuron models in the literature.

Our analysis shows that there are significant differences in the performance profiles of *in silico* models falling within the same category of cell-based representations. Features related not only to the neuron abstraction but also to the scientific question under analysis can cause variations in the hardware bottlenecks and ultimately influence simulation performance. We find instances where the level of morphological detail is not a factor in distinguishing between models' performance, while the synaptic formalism is, allowing us to identify the key factors that determine a model's computational profile. Finally, we show that our analysis is strongly conditioned on the published models' parameters and simulation dynamics, predicting how changes in some values, notably the firing frequency, can significantly alter the performance profile. Short of being able to run benchmarks on different kinds of hardware architectures, we use our model to explore the effects of machine balance and hardware design choices on the performance of *in silico* brain tissue simulations, providing actionable guidelines for hardware procurement and co-design. Ultimately we come to the conclusion that while general-purpose computing has, until now, largely been able to deliver high performance, the next generation of brain tissue simulation will be severely limited by hardware bottlenecks. Indeed this trend was already foreshadowed in empirical studies (Jordan et al. 2018), and solutions involving hardware accelerators such as General Purpose Graphical Processing Units (GPGPUs) have been proposed (Fidjeland et al. 2009; Yavuz et al. 2016; Brette and Goodman 2012), while the development of custom brain-like hardware is being actively explored with promising results (van Albada et al. 2018; Wunderlich et al. 2018).

We believe that the situation calls for a better, deeper understanding of how hardware capabilities interact with brain tissue simulation algorithms. In turn, this would allow a stronger collaboration between *in silico* modelers, developers and hardware specialists to orchestrate the co-design of software and hardware architectures. The methodology developed in this work constitutes a quantitative means through which these scientific communities can collaborate in the task of designing and optimizing future software and hardware for the next generation of brain tissue simulations.

Related Work

As a testament to the growing interest of the community in the performance of brain tissue simulations, several studies on this topic have been published in the literature. To our knowledge, however, none of them have used performance modeling as a tool to explain the empirically observed performance properties of simulations, nor have they tried to analyze such a wide scope of models as we do in this paper.

The review work of Brette et al. (2007) presented a large number of different simulators and corresponding *in silico* models, but included only basic formulas for asymptotic complexity, without exploring the complicated effect that implementation and hardware have on measured performance. In a series of papers, the developers and users of the NEST software have investigated the issues related to scaling simulations of neurons to very large scales (Kunkel et al. 2014; Peyser and Schenck 2015; Ippen et al. 2017; Jordan et al. 2018) and proposed solutions to avoid the performance bottlenecks they encountered. Similarly, new simulators and spike communication strategies have been explored (Ananthanarayanan and Modha 2007; Kozloski and Wagner 2011; Hines et al. 2011). All these studies provide very useful data to compare against our own model and conclusions, but are restricted to distributed simulations and focus on optimizing efficiency using novel algorithmic and implementation techniques. A performance model for a NEST simulation was fully described in Schenck et al. (2014), but focuses only on distributed simulations, neglecting single-node performance, and uses a different performance modeling approach based on interpolating empirical observations instead of the semi-analytical approach used in this work. Focusing on very small clusters composed of only a few nodes with shared-memory capability, Eichner et al. (2009) demonstrated how to exploit multicore processor efficiently in simulations of morphologically detailed neurons, while an analysis concluded that real-time simulations of medium sizes plastic networks are not feasible on small clusters of CPUs, but will require accelerators or dedicated hardware (Zenke and Gerstner 2014). The work on the multisplit method (Hines et al. 2008) demonstrated that efficient acceleration of individual neurons on single compute nodes is difficult beyond a restricted number of cores, and recent work on micro-parallelism (Magalhaes et al. 2019a; 2019b) has found significant limitations to strong-scaling due to Amdahl's law.

Several GPU implementations of brain tissue simulations have been proposed in the literature (Brette and Goodman 2012; Fidjeland et al. 2009; Yavuz et al. 2016; Kumbhar et al. 2019b). A comparison between GPUs, HPC hardware and neuromorphic hardware found that, under certain conditions, GPUs can beat neuromorphic hardware in terms of energy efficiency but not an HPC server in terms of performance (Knight and Nowotny 2018). However, the authors did not perform a detailed performance analysis nor used a performance model to explain this comparison, thus providing valuable yet anecdotal evidence. Their work is tightly linked to the neuron model, the configuration of the simulation as well as the hardware being analyzed.

In the process of creating the neuromorphic hardware SpiNNaker, the designers were deeply interested by the consequences of their decisions in terms of performance. An analysis showed that, in the context of real-time simulations, SpiNNaker is inherently limited in its scope to a restricted subset of synaptic formalism, because the single node's memory bandwidth puts a hard limit on the number of synaptic parameters that can be streamed at every timestep (Painkras et al. 2013). In designing the network for SpiNNaker, a performance model was used to determine the ideal topology and network connectivity (Navaridas et al. 2012), proving that performance modeling can be an extremely valuable tool in optimization and co-design for brain tissue simulations.

Materials and Methods

We develop our analysis of the performance landscape in a two-step process as show in Fig. 1: first we identify relevant *in silico* models and experiments from the literature that constitute a representative sample of state-of-the-art models and algorithms, and present a set of hardware-agnostic descriptive metrics that can give a first insight on their performance properties; then we intersect the hardware-agnostic description with a model of the hardware platform, by extending and adapting well-established performance models to neuroscientific simulations use cases.

In Silico Models and Experiments

The approach of identifying and singling-out recurrent computational patterns within a scientific field has been applied with great success in the domain of parallel computing, leading to the definition of the dwarfs of computing (Asanovic et al. 2009). In computational neuroscience, the review by Brette et al. (2007) proposed a similar approach and introduced some fundamental concepts, such as conductance based (G-based) and current based (I-based) formalisms for synaptic models, or point and detailed representations of neuronal morphology. Using the nomenclature introduced in that review, we base our analysis of the performance landscape on three published models, chosen as representative of the extent of neuron models covered in literature. We denote the set of these representative use cases as *in silico models and experiments*, and summarize their salient properties in Fig. 2a.

The *Brunel* model is a randomly connected network reproducing the property of balanced excitation and inhibition that can be observed in the brain cortex. It is based on integrate-and-fire (IAF) point neurons with I-based synaptic dynamics (Gerstner et al. 2014; Brunel 2000). As a representative example of this model, we consider here a very large-scale implementation that served as a proof of concept for the feasibility of human brain scale simulations (Kunkel et al. 2014).

The *Reconstructed* microcircuit is based on the reconstruction of neocortical microcircuit from a mix of experimental data and first principles (Markram et al. 2015). This

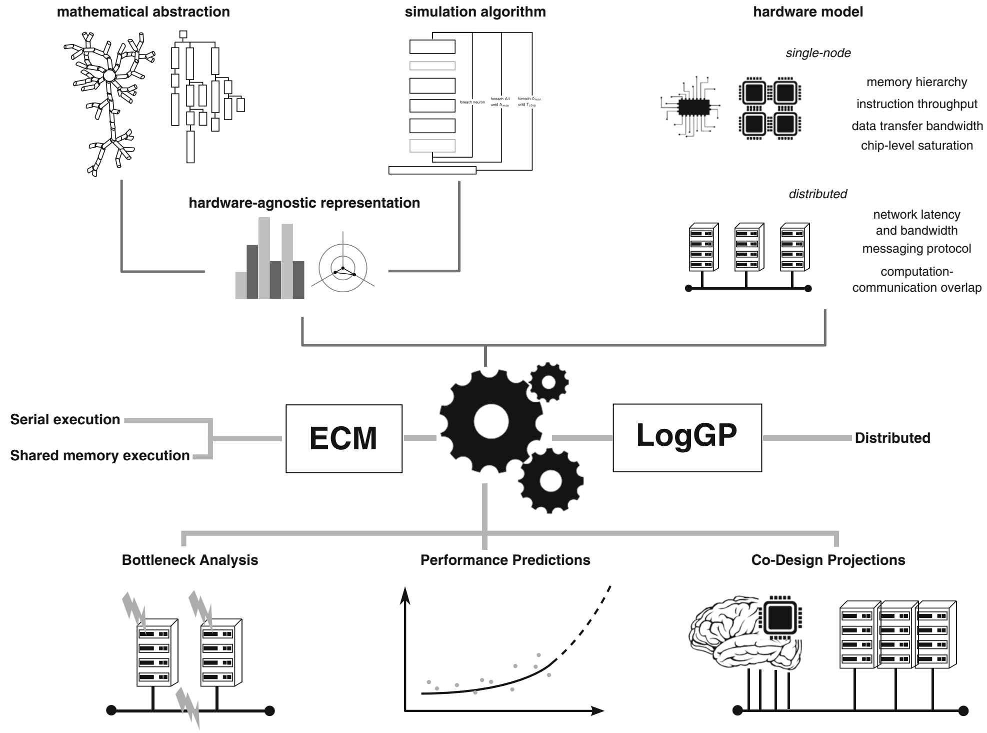

Fig. 1 Comprehensive performance modeling of brain tissue simulations. An hardware-agnostic representation of the *in silico* model is obtained by combining detailed information about the mathematical abstraction, such as e.g. the representation of neurons and its implementation as data structures, or the formulation of the differential equations at the basis of the temporal and spatial dynamics, as well as the simulation algorithm and the dependencies between different simulation phases. This is combined with an abstract representation of the hardware based on a few key parameters, as well as a detailed understanding of the software implementation and the execution of the flow of instructions on the reference hardware, to obtain runtime predictions based on the ECM model for serial and shared-memory execution, and on the LogGP model for interprocess communication. Once our performance model is validated, we use it to predict the performance of brain tissue simulations in multiple configurations, analyze bottlenecks through introspection of the model and provide informed guidelines for the co-design of future hardware

model uses morphologically detailed neurons and short-term plastic G-based synapses to capture a large amount of biological detail.

The *Simplified* model was obtained by reducing the detailed models of the Reconstructed microcircuit to pointwise generalized integrate-and-fire (GIF) models with similar transfer functions, while retaining the complexity from G-based synapses (Rössert et al. 2016).

The scope of this paper is restricted to the inference phase of simulating a neural network, thus neglecting the simulation of learning, because of the additional layer of complexity that would arise from including long-term plasticity in our analysis. Moreover, to allow for reproducibility and ease of modeling, we have switched the random synaptic release mechanism in the Reconstructed and Simplified models with a deterministic implementation of the same rule, based on an average representation. This allows us to remove the uncertainty in the performance model due to probabilistic release, as well as the overhead from the random number generation, ultimately leading to better accuracy in the performance model. Although synaptic noise can be considered a primary driver of cortical dynamics (Nolte et al. 2019), we consider the complexity associated with random number generation in performance modeling outside of the scope of this paper.

Simulation Algorithm We identify similarities and differences between the simulation algorithm of the *in silico* models and experiments. The first important remark is that the phases that make up the complete algorithm depend only on the type of synaptic formalism, and not on the level of morphological detail in the neuronal abstraction. Thus in Fig. 2a we present two algorithm skeletons: one for I-based models and one for G-based models. Moreover, regardless of the *in silico* model, there are always three nested loops: the outer one determines when global synchronization and communication of spike events happens; the central loop iterates over the simulation timesteps between two global synchronization events; the inner loop updates the state of each neuron by one timestep. In the I-based simulation workflow one needs, in addition to the event-driven integration of synaptic events, only to update the voltage of

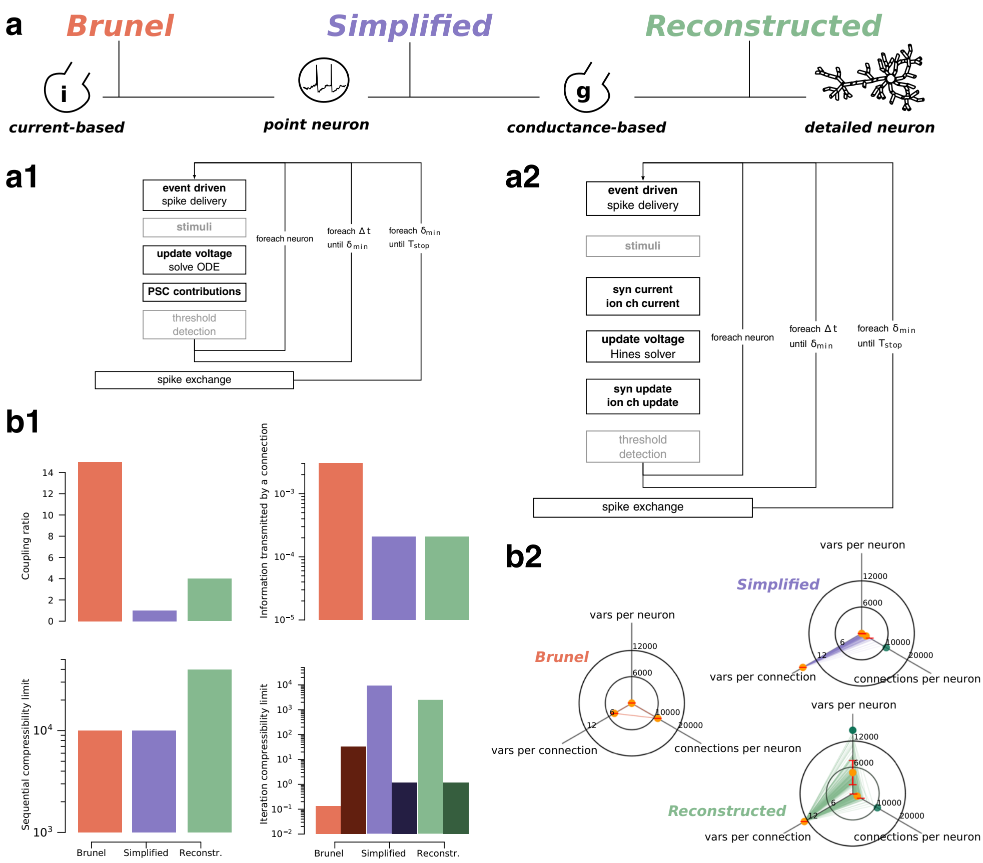

Fig. 2 In silico models and experiments. Presentation and summary of the *in silico* models and experiments examined in this paper. **a** Color-coding for the three *in silico* models and salient features: in red the I-based point neuron *Brunel* model, in purple the G-based point neuron *Simplified* model and in green the G-based detailed neuron *Reconstructed* model. **a1,a2** I-based (resp. G-based) simulation algorithm. The simulation kernels within light grey boxes are included for completeness but are not considered in our analysis because they are not part of the computation loop. The larger boxes denote a synchronization point for distributed simulations. **b1** Hardware-agnostic metrics. Coupling ratio denotes the number of simulation timesteps before a global synchronization point. Information transmitted by a connection denotes the average number of variables transmitted via a connection during one minimum delay period. Sequential compressibility limit denotes the number of time iterations required to simulate one second of biological time. Iteration compressibility limit denotes the number of degrees of freedom updated in a δ_{min} interval. Lighter bars represent clock-driven updates, darker bars represent (average) event-driven updates. **b2** Breakdown of the unit size metric. This metric captures the memory footprint of *in silico* models, broken down in three components: number of variables to represent a single neuron excluding synaptic connections, number of variables to represent a connection, and number of connections per neuron. Orange dots represent mean values, red bars represent standard deviation and green dots represent maximal values. The lines represent actual samples from the model

individual neurons by solving a few simple ordinary differential equations (ODE) and to update for each neuron the total post-synaptic current (PSC) contributions. In the G-based model, a linear system of equations must be solved for each neuron and each timestep to updated the voltage. In addition, for every synapse and ion channel one must compute its corresponding contribution to the matrix in the so called *current* kernels as well as update its state in the state kernels.

Hardware-Agnostic Metrics to Describe *in silico* Models and Experiments As a first approach to navigating the performance properties landscape of *in silico* models and experiments we propose a collection of hardware-agnostic metrics that can be computed directly from inspection of the model specifications. These metrics have the same value regardless of the underlying simulation hardware, an interesting property if one wants to compare *intrinsic* model features. To set a common ground on which we define these metrics we identify the following features shared by all *in silico* models and experiments: i) all neural networks can be represented by a graph where neurons are nodes and synaptic connections are edges; ii) synaptic connections can be approximated by a perfect delayed transmission of information; iii) simulation algorithms include a clock-driven portion to integrate neuronal states and an event-driven portion to integrate synaptic events; iv) all simulation algorithms considered here follow the Bulk Synchronous

Parallel (BSP) paradigm, where computation phases carried out independently by each parallel rank alternate with global synchronization steps, happening at fixed time intervals called minimum network delay (denoted by δ_{min}). We are thus excluding from this analysis asynchronous communication schemes (Ananthanarayanan and Modha 2007; Magalhães and Schürmann 2019), variable timestep schemes (Lytton and Hines 2005) and models that explicitly represent axons (Kozloski and Wagner 2011).

We evaluate the performance metrics of *in silico* models on three aspects: memory, serial complexity and information propagation. In the memory dimension we consider aspects of the model that can affect the memory capacity footprint such as the number of parameters and degrees of freedom required to represent a neuron. Specifically we count the number of state variables and parametes per neuron, the number of state variables and parameters per connection and the fan in (i.e. the number of incoming connections) per neuron. In the serial complexity dimension we consider aspects tied to sequential iterations such as timestep and the number of state variables updated at each iteration. We define the *sequential compressibility limit* as the inverse of the timestep Δt and the *iteration compressibility limit* as the number of state variables updated in a single time iteration. Finally, in the information propagation dimension, we consider aspects tied to communication of information between neurons such as the frequency at which a global synchronization must happen and the amount of information exchanged in this step. Here we define the *coupling ratio* as the number of timesteps that can be taken before a global synchronization point must happen, given by the formula $\frac{\delta_{min}}{\Delta t}$, and the *information transmitted by a connection* as the number of variables communicated by a connection on average during a minimum network delay period.

Figure 2b reports the values of the hardware-agnostic performance metrics obtained by hand-counting the relevant quantities in the published *in silico* models. Each of the metrics described above can be associated to one or more performance aspects and hardware features. For example, the low values of the coupling ratio for the Simplified and Reconstructed model can be associated to poor strong scaling properties, while the large information transmitted by a connection for the Brunel model translates to higher pressure on the network interconnectivity hardware. Concerning time iterations, the large event-driven component of the iteration compressibility limit for the Brunel model points to the fact that it could potentially be bounded by hardware latency aspects (either memory latency or critical paths in the execution) as well as dynamic imbalance, while the Simplified and Reconstructed model are more likely affected by throughput of hardware features. The large number of connections per neuron in the Brunel model and the large number of variables to represent a neuron in the Reconstructed entail that these *in silico* models will be bounded by memory capacity. Finally, the large variability of individual neurons in the Reconstructed model poses a potential static load-balancing problem, as was empirically found in Kumbhar et al. (2018) in the context of manycore processors.

Analytical Performance Modeling of Brain Tissue Simulations

The metrics described previously provide an insightful summary of the performance profile of *in silico* models and experiments, but lack the power to give quantitative performance predictions and the connection with specific hardware properties. Therefore, we use performance modeling as a way of bridging the gap between biophysical models, simulation algorithms and hardware specifications. In particular, we split the performance prediction in a single-node component and an interprocess communication component. We address the single-node performance modeling using the Execution-Cache-Memory (ECM) model (Treibig and Hager 2010) and the interprocess communication part using the LogGP model (Alexandrov et al. 1997). Both are well-established approaches that have been extensively validated on several hardware platforms (Hager et al. 2016, 2018; Hoefler et al., 2009), however significant work is required to extend and adapt them to the simulation kernels in our analysis, for example accounting for indirect memory accesses in the single-node predictions and the representation of spikes in the communication component. Details for the extension and validation of the ECM and LogGP performance models are provided in the Supplementary Material S1.1 and S1.2.

Single-Node Performance Model The ECM model uses a *grey-box* mixed approach combining an analytic formulation with some phenomenological input, and outputs a runtime prediction at the granularity of individual clock cycles (Treibig and Hager 2010). Since its introduction it has been refined and validated on modern Intel and AMD multicore architectures (Hofmann et al. 2017, 2018; Stengel et al. 2015) . To compute the ECM performance model for serial execution one must first define several contributions to the runtime of a given loop, such as: the in-core execution time assuming data is already loaded in registers T_{OL}, the time needed to load data into registers from the L1 cache T_{nOL}, the data traffic time between caches T_{L1L2}, T_{L2L3} and the data traffic time from main memory T_{L3Mem}. Data traffic times are usually computed combining an estimation of the data traffic with the bandwidth of the relevant data link. T_{OL} and T_{nOL}, on the other hand, can be computed by hand but are typically extracted using code analysis tools such as Intel's IACA (Intel 2017). These contributions must be combined to obtain two quantities: T_{core} and T_{data}, representing the time that the loop would spend in core execution if data were instantaneously available, and the time required to move the data across the memory hierarchy, respectively. One of the core assumptions of the ECM model is that these two quantities can overlap, therefore single-thread runtime predictions can be obtained using the formula

$$T = max\left(T_{core}, T_{data}\right). \quad (1)$$

The ECM model is based on the full-throughput assumption, thus neglecting any latency effects in the execution. This assumption greatly simplifies the analysis by removing the need for an extremely detailed understanding of the execution flow while at the same time providing *insight through failure* for situations in which the program execution is the bottleneck. In this context, a particular kernel will be categorized as **core-bound** if $T_{core} > T_{data}$, and **data-bound** otherwise. Note that these definitions apply to the serial execution. To obtain a performance prediction for parallel execution, the ECM model assumes that performance scales linearly with the number of threads, until a bottleneck from a shared serial resource is used, typically the memory interface (Hofmann et al. 2015). The ECM also provides a formula for computing the **saturation point**, i.e. the number of shared memory threads at which saturation of the memory bandwidth occurs for a given kernel.

We computed the individual ECM dimensions and the corresponding runtime predictions for all clock-driven kernels of the *in silico* models, and report them in Supplementary Table S4. Moreover, we conducted a thorough validation of our runtime predictions and report the results in Supplementary Table S5 and Supplementary Fig. S10.

Spike Delivery Kernel The performance modeling of the spike delivery kernel presents several challenges that warrant the need for a separate treatment. In terms of algorithm design, all state-of-the-art software use some sort of priority queue or ring buffer to store synaptic events to be delivered within a timestep. For modeling and benchmarking, we separate the operations related to the bookkeeping of events inside the queue from the actual kernel execution, and we only consider the latter, because the scope of our analysis is restricted to computational and communication kernels.

When a spike is received, the postsynaptic process must integrate its effects in the state of the target neuron or synapse. In I-based synapses this amounts to increasing a spike counter by the relative weight of the connection, while for G-based synapses an equivalent quantal update of the synaptic states must be computed, a computationally expensive procedure since the *in silico* models considered in this work all contain a short-term plasticity model.

The spike delivery kernel is characterised by erratic memory accesses, because the order of activation of synapses is unpredictable. We always consider the worst possible case in which every spike to be delivered could not be cached and thus must come from main memory. We assume that a full cache line of data needs to be brought in from memory *for every data access*, since the unpredictable order of activation of synapses renders data prefetching and data blocking largely ineffective. We consider that only accesses to synapse-specific data are non-contiguous and thus require a full cache line (64 B) of data to be transferred for every memory request.

Estimating the runtime proves to be a very challenging task. We find that the naïve approach of multiplying the DRAM latency by the number of non-contiguous accesses yields very pessimistic predictions. This can be attributed to the fact that, since spikes are independent, it is not necessary to wait until one spike has been processed before issuing request for the data of the next spike. It thus seems that the spike delivery kernel's performance is determined by the number of concurrent, independent data requests that can be handled by the processor and memory. This is different from the classical purely latency bound kernels in which the CPU is only allowed to begin a loop iteration after the previous one is fully completed. The number of independent memory requests that can be handled concurrently is known as memory level parallelism (MLP) and allows to mitigate the performance impact of memory latency by allowing multiple accesses in parallel (Levinthal 2014). For shared memory parallelism, we assume that performance scales linearly with the number of threads until the bottleneck of memory bandwidth is reached. Details about the runtime prediction and validation are provided in the Supplementary Material S1.1 and Supplementary Fig. S11.

Spike Exchange in Brain Tissue Simulations All the *in silico* models and experiments considered in this paper are based on the Bulk Synchronous Parallel (BSP) model (Valiant 1990), which prescribes a clear distinction between an *on-node computation* phase (happening in a distributed parallel fashion) and an *inter-node communication* phase. For brain tissue simulations, the inter-node communication phase corresponds to the spike exchange step in Fig 2a. Moreover, we make the assumption that the distributed processing is implemented in MPI, because it represents the current state of practice in the HPC community. Throughout this work, we maintain the nomenclature of shared memory threads and distributed ranks. When we use the generic term of parallel processes, we make the assumption that shared memory parallelism capabilities are always exhausted before distributed memory parallelism.

Details on our application of the LogGP model to the spike exchange algorithm, as well as on the reference hardware, are provided in the Supplementary Material S1.2.

In all the state-of-the-art simulators, the spike exchange step is implemented by a blocking collective call, typically a variant of the Allgather operation. This entails that all the parallel ranks have, at the end of the communication step, knowledge of all the spikes produced by the simulation during the last minimum network delay period. Recent work has shown that at extremely large scales, this implementation can become prohibitively expensive in terms of memory requirements, and proposed to use instead the Alltoall operation to deliver spikes only to the ranks where they are required (Jordan et al. 2018). Other alternative implementations have been suggested, using nonblocking point-to-point communication (Ananthanarayanan and Modha 2007) or spatial decomposition (Kozloski and Wagner 2011). All these fall outside of the scope of this paper, which is focused on small-to-medium cluster sizes and well established, widely used software solutions.

The LogGP Model for Interprocess Communication We use the LogGP model (Alexandrov et al. 1997) to predict and explain the performance of the spike exchange simulation step. The LogGP model is an extension of the LogP model (Culler et al. 1993) that uses an additional parameter, the gap per byte denoted G, as a way to account for the sending and receiving of long messages. We refer the interested reader to the Supplementary Material S1.2 for more details. The main features of all models based on LogP is that their parameters are easily relatable to hardware characteristics, thus ensuring a high degree of interpretability. In the LogGP model, the cost of sending a single message of size m bytes is given by two contributions: a latency contribution corresponding to the time it takes for the first byte of the message to reach its destination, and a bandwidth contribution corresponding to the throughput at which messages can be communicated through the interprocess network. One of the main insights in the LogGP model is that, under certain circumstances, CPU-side operations such as copying of data can overlap with network-side operation such as data sending.

Collective communication operations are difficult to model because different algorithms can be used to disseminate the messages across the network, and the choice of which one to use can happen dynamically and transparently to the user, depending on several performance factors. A review mentions however that the ring algorithm is the most commonly used, especially for large messages, and thus we base our predictions on this paradigm (Thakur et al. 2005). In the ring algorithm, the number of times in which parallel processors establish a network connection and start exchanging data is proportional to the total number P of parallel ranks in the simulation, therefore the latency term of the total cost of the collective communication is expected to scale linearly in P. Conversely, in the recursive doubling algorithm this term scale logarithmically, while for both algorithms the total amount of communicated datais the same, and thus the bandwidt term for both algorithms is expected to be the same. Using this information, we adapt the formula proposed originally by Mamadou et al. (2006) for recursive doubling, to account instead for the ring algorithm, as explained in equation (S10) in the Supplementary Material. In our modeling and validation we always consider that the number of spikes communicated by each rank is roughly homogeneous. The validation of our interprocess communication runtime predictions are provided in the Supplementary Fig. S12.

Hardware Models

Our performance modeling methodology requires a detailed abstraction of the hardware to obtain accurate runtime predictions. In this work we use the Intel(R) Xeon(R) Gold 6140 Skylake processor as reference for the single-node hardware, We provide in Supplementary Table S3 the values of the hardware characteristics that are most relevant for our performance modeling methodology. while for distributed communication we use as reference a vendor (HPE) provided MPI implementation, based on MPT 2.16 and the MPI 3.0 standard, over an Infiniband EDR 100 GB/s fabric. This architecture is highly representative of modern, state-of-the-art high performance computing clusters of CPUs. For validation purposes, we execute benchmark simulations on the reference hardware by inputting random synaptic inputs at a frequency of 1Hz into every cell within a network of disconnected neurons. All the benchmarks were executed multiple times under the same conditions (typically around 10 runs), and we define the error (or margin of error) as the ratio of the difference between the median measurement and predicted runtime divided by the median measurement. For validation and benchmarking we use the CoreNEURON implementation as reference (Kumbhar et al. 2019b).

Using our reference Skylake hardware as basis, we also develop the concept of strawman architectures: models of the hardware that do not necessarily reflect reality perfectly in every detail, but that capture the most salient hardware properties and can be used to explore the design space easily. We consider in this work another multicore CPU server based on the AMD Naples architecture, a manycore Intel Knight's Landing (KNL) architecture and a GPU-like architecture inspired by the Nvidia Volta V100 GPU. For each strawman architecture, we took the amount of shared-memory parallelism, the memory bandwidth, the clock frequency and the cache hierarchy from nominal values or published studies (Hofmann et al. 2019; Jeffers et al.

2016; Jia et al. 2018). Other important features such as the instruction throughput and the memory level parallelism and latency, that require extensive benchmarking, could not be obtained directly. Instead, we opted to infer them from the corresponding values in the reference Skylake architecture, taking the memory level parallelism and latency as they were, and scaling the instruction throughput to the appropriate level of vectorization. For simplicity, we considered only the high-bandwidth memory on the KNL architecture, and we treated individual streaming multiprocessors in the GPU-like architecture as shared memory threads with SIMD registers of 32 double-precision floating point variables width, connected directly to the global GPU memory.

Results

Our goal is to provide a quantitative appraisal of the performance landscape of brain tissue simulations and analyze in detail the relationship between an *in silico* experiment, the underlying neuron and connectivity model, the simulation algorithm and the hardware platform being used. We carry out this analysis with the tool of performance modeling, allowing us to quantify and explain performance bottlenecks without the need of time-consuming and narrowly-scoped benchmarks.

Serial Performance Profile

We uncover the fundamental performance properties of simulations of biological neurons by examining the serial performance profile. We counted by hand the number of floating point operations (flop) and the data traffic required to simulate one neuron receiving synaptic events at a rate of 1Hz for one second of activity, for different modeling abstractions. Table 1 summarizes the results. As expected, the flop and data requirements grow with the amount of biological detail in the model, leading to a factor 10^4 difference between the Brunel and Reconstructed model. Interestingly, the arithmetic intensity of all the models is roughly the same, in the order of 10^{-1}. This may be ascribed to the fact that the arithmetic intensity of the differential equations being solved in each model is essentially very similar, and only the number of equations to represent a neuron within each modeling abstraction is different. We also looked at the total amount of memory required to store a neuron's states and parameters, and found that the Simplified model's strategy of lumping synapses allows it to greatly optimize the memory capacity requirements. Finally, we conclude the analysis by using our model to predict the simulation performance, measured in biological seconds per elapsed wallclock second to simulate a single neuron receiving synaptic events at a rate of 1Hz. A factor 10^4 difference in the performance of the Brunel and Reconstructed model mirrors perfectly the difference in flop and data traffic requirements observed above.

Table 1 Average number of flop and data traffic per neuron to advance its state by 1 second of simulated time

	Brunel	Simplified	Reconstructed
Flop	2.3×10^5	1.7×10^7	8.5×10^9
Data Volume [MB]	1.0	1.3×10^2	3.1×10^4
Arith. Int. [Flop/B]	2.3×10^{-1}	1.3×10^{-1}	2.7×10^{-1}
Mem. Capacity [KB]	2.7×10^2	10	2.9×10^3
Perf. [sim s / wall s]	3.4×10^3	1.7×10^2	3.4×10^{-1}

Arithmetic Intensity is defined as the ratio of flop per data volume. We also report the predicted serial simulation Performance, measured in simulated seconds per elapsed wallclock second for a single neuron receiving synaptic events at a rate of 1Hz. The requirements in this Table are computed considering only the data structures strictly relevant to computation, thus neglecting overhead from implementation details such as MPI buffers, data structure representation, memory padding, etc

We complete our above analysis with a detailed characterization of the performance profile of simulation kernels based on the reference hardware. At first we consider the T_{core} and T_{data} components of all the individual kernels that constitute the simulation algorithm of different modeling abstractions. Figure 3a demonstrates that most kernels are data-bound on the reference architecture, even when in the case of short vector registers (SSE). When larger vector registers are used (i.e. AVX512), the T_{core} component can be greatly improved while the T_{data} remains roughly the same. This is an indication that the performance of core-bound kernels can be improved by vectorization, until the kernel becomes data-bound. In addition, we observe that when using the vectorization hardware to its fullest potential (i.e. AVX512) several kernels lie on the boundary between core-boundedness and data-boundedness. This represents a balanced profile where the ratio of data traffic and computation matches the design space of the hardware. To draw conclusions about the full models, we need to intersect this information with the relative importance of the individual kernels on the total runtime. The serial performance of G-based models is dominated by the state and current kernels, in roughly equal parts. Thus we conclude that G-based models are mainly data-bound. In the I-based model the most time consuming kernel is the event-driven spike delivery. This implies that, while the clock-driven portion of the I-based model is definitely data-bound, the serial performance hardware bottleneck of the whole neuron model is memory-level parallelism and latency.

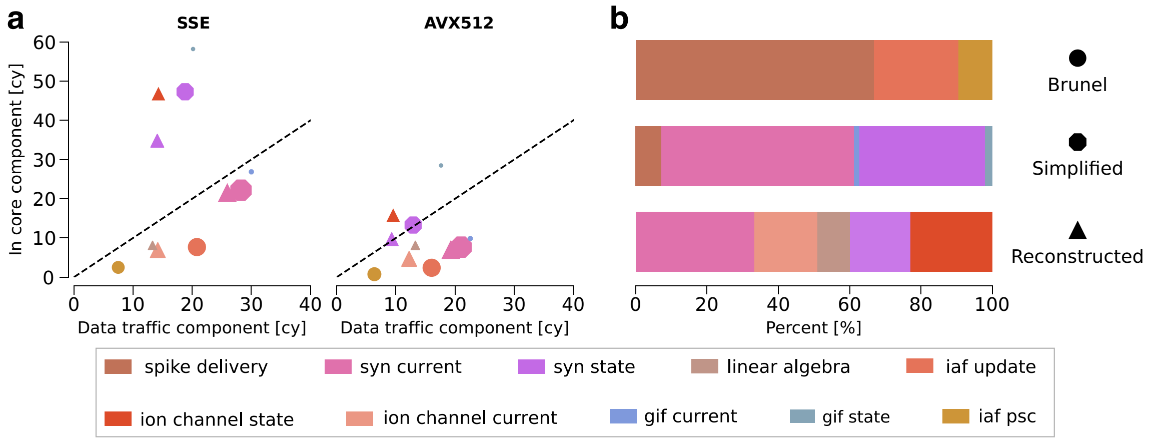

Fig. 3 Predicted serial performance characteristics of clock-driven computational kernels in brain tissue simulations. We predict the serial runtime of *in silico* models as a sum of their individual kernels on the reference SKX AVX512 architecture. **a**: T_{core} and T_{data} components of the clock-driven kernels from brain tissue simulations. The dashed black line delineates the boundary between core-bound kernels (over the line) and data-bound kernels (under the line). Marker type denotes the *in silico* model whence the kernel is taken, while marker size is proportional to the relative importance of the kernel in the total runtime. **b**: breakdown of the relative importance of individual kernels over the total serial runtime

Memory Bandwidth Saturation in Shared-Memory Execution

One of the most common simulation configurations involves scaling the number of neurons until the memory capacity limit is reached. This configuration has been used as proof-of-concept for brain tissue simulations to the scale of brain regions and even the full brain and constitutes a fundamental tool for neuroscientists to simulate networks whose sizes are representative of the neural systems they are studying (Ananthanarayanan et al. 2009; Jordan et al. 2018; Izhikevich and Edelman 2008).

Memory Bandwidth Limits Shared-Memory Parallelism Modern architectures are typically designed with memory bandwidth as the most relevant bottleneck for shared-memory parallelism (McCalpin 1995). This means that if all the shared memory parallel threads are used, it is very likely that performance will be bounded by the memory bandwidth. Indeed, this has been demonstrated to be the case for simulations of detailed neurons (Cremonesi et al. 2019) and strongly suspected in the case of point neurons (Zenke and Gerstner 2014). To verify this hypothesis we compute the memory bandwidth utilization for the three *in silico* models considered in this work. The results are shown in Fig. 4a. We find that all models pass the threshold of 90% utilization well before all available parallel threads are utilized, meaning that memory bandwidth is indeed a bottleneck under the assumption that data must be pulled from main memory *at every time iteration*. However, we surprisingly also find that, regardless of the level of morphological detail, G-based models share a similar pattern of early saturation while the I-based IAF model requires slightly more parallelism to achieve memory bandwidth saturation. In G-based models this is explained by the dominance of synaptic and ion channel current kernels which determine the early saturation pattern, whereas in the I-based model the saturation is driven by the memory latency effect on the spike delivery kernel.

State-of-the-Art HPC Memory Chips Can Sustain Fast Simulations of the Brunel and Simplified Models at Full Saturation From a practical point of view, in addition to analyzing the scaling behaviour of simulations, computational neuroscientists are also interested in predicting the actual runtime for a given model. Thus we predict that the simulation performance, under the assumption that *memory bandwidth is fully saturated*, is 1.6×10^4, 7.9×10^2 and 1.7 simulated seconds per wallclock second per neuron, for the Brunel, Simplified and Reconstructed model respectively. Our results indicate that the modern, fast memory chips on the reference architecture are able to sustain faster-than-real time simulations of up to roughly 10^4 neurons in the Brunel model, and 10^3 neurons in the Simplified model, while faster-than-real time simulations of the Reconstructed model on the reference hardware are predicted to be theoretically possible only by a narrow margin, and in practice probably impossible. Performance predictions under the memory bandwidth saturation assumption represent a theoretical upper limit on the achievable performance through shared memory parallelism. The following paragraph explains how this limit can be overcome through algorithmic improvements.

Ordering of Loops to Avoid Memory Bandwidth Saturation State-of-the-art simulators employ a specific ordering of the

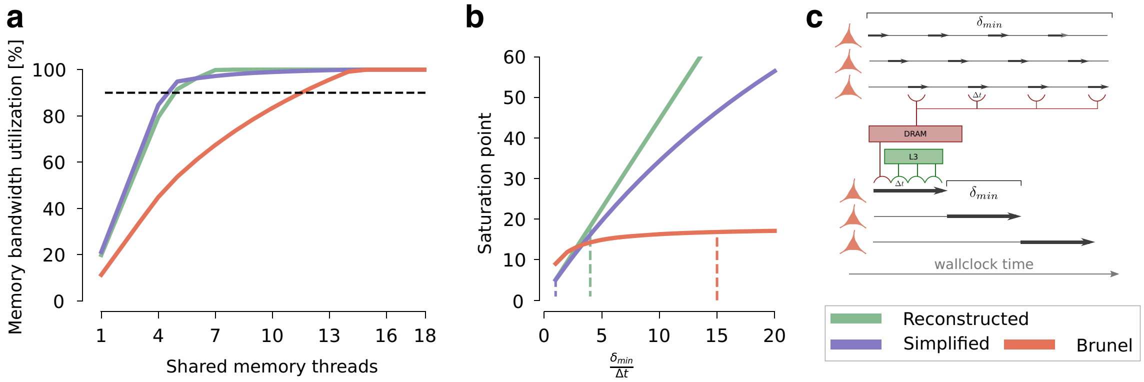

Fig. 4 Predicted shared-memory performance characteristics. We predict the shared-memory runtime of *in silico* models as a sum of their individual kernels on the reference SKX AVX512 architecture. **a** Percentage of memory bandwidth utilization as a function of the number of shared memory threads. The dashed black line denotes the threshold of 90% utilization. **b** To mitigate the effect of memory bandwidth saturation, a smart ordering of time and neuron loops is implemented by state-of-the-art simulators, as shown in the diagram on the right. We plot the number of threads required to reach saturation of memory bandwidth as a function of the coupling ratio. Different coupling ratios were enforced by keeping the Δt fixed to each model's published value, and changing the $\delta_{\min}$ accordingly. Dashed lines represent the actual published values for the coupling ratio. **c** schematic representation of the loop ordering optimization to improve cache reuse. The top shows the naïve implementation: each neuron, represented by an horizontal line, is advanced by a single timestep, as shown by the short black arrows. In this case, every time a neuron's state is advanced by one timestep data must be fetched from the main memory (red lines), since the caches will be overwritten by the data from other neurons at the same timestep. The bottom shows the optimized version: each neuron is advanced by several timesteps (longer black arrows) until it reaches a $\delta_{\min}$ boundary. In the optimized version data must be fetched from main memory only during the first timestep, while consequent operations can reuse the data for the same neuron immediately (green lines represent data coming from the L3 cache)

loops over neurons, timesteps (Δt) and minimum network delay steps ($\delta_{\min}$) to minimize the impact of memory bandwidth by maximizing data locality. This optimization is summarized in Fig. 4c. Throughout this work we make the conservative assumption that, when using the loop ordering optimization, data must be fetched from main memory on the first timestep and from the L3 cache on consecutive timesteps until a minimum delay barrier is reached. The number of timesteps within a minimum delay period has of course a great influence on the effectiveness of this strategy in terms of reducing pressure on the memory bandwidth. To quantify this, we compute the number of threads to reach saturation – n_{satur} – and plot the results in Fig. 4b as a function of the coupling ratio defined by $\frac{\delta_{\min}}{\Delta t}$. In G-based models, there is an almost linear relationship between the coupling ratio and n_{satur}, indicating that investigating ways to increase the coupling ratio could be highly beneficial for parallelism. Note that, in this regard, increasing the coupling ratio by decreasing Δt presents a performance tradeoff: it allows more parallelism but increases the computational requirements (number of iterations) of the model. Conversely, while the minimum network delay is obviously a fixed parameter of the network that cannot be arbitrarily changed, methods that experimented with a per-neuron delay, instead of a network-wide minimum delay, demonstrated significant speedup (Magalhães and Schürmann 2019). The relationship between coupling ratio and n_{satur} for I-based models is bounded by a relatively small limit of roughly $n_{satur} \leq 17$, above which no additional parallelization is predicted to provide any benefit. This is explained by the fact the spike delivery kernel, in virtue of its event-driven nature, is unaffected by the benefits of the coupling ratio. Since our assumption is that data for this kernel must always come from main memory, as soon as it becomes the dominating performance factor and it reaches saturation, it inhibits any benefit from parallelism.

High Speed Single-Node Simulations

Another widespread simulation regime is focused on simulating a fixed size network as fast as possible. We call this the constant problem size regime, and within it we make the assumption that the optimized loop ordering is always implemented to minimize the pressure on the memory bandwdith. One use case falling within this performance regime is real time simulations, in which one second of simulated time requires at most one second of wallclock time. Currently, on the one hand it is unclear whether real time is realistically achievable on modern hardware (Zenke and Gerstner 2014), and on the other hand special hardware that breaks this limit by design has already been conceived and tested for small networks (Aamir et al. 2018).

Memory Bandwidth Dominates the Shared-Memory Strong Scaling of Brunel and Simplified Models, while a Mix of Hardware Features Influences the Performance of the Reconstructed Model Simulations of networks comprising

Table 2 Predicted performance per neuron without the memory bandwidth saturation assumption, considering all available threads are used

	Brunel	Simplified	Reconstructed
performance (DRAM)	5.2×10^4	7.9×10^2	3.9
speedup in L3	2.5	4.7	1.8
speedup in L2	8.7	6.5	2.4
speedup in L1	33.9	6.6	2.4

Performance is measured in simulated seconds per wallclock second

a small number of neurons can be advantageous because if the dataset can be fully contained in the CPU caches, superlinear speedup can be observed. Therefore we predict the simulation performance per neuron assuming the dataset could be fully contained in different levels of the memory hierarchy. For simplicity, we neglect the fact that some of these model and cache combinations are infeasible in practice, e.g., due to the memory footprint of a single neuron in the Reconstructed model exceeding the L1 cache size. The performance predictions are reported in Table 2 assuming all available threads in the reference architecture (18 in total) are being utilized. Note that the reported performances are better than the theoretical limit computed in the previous section by assuming memory bandwidth saturation. While this may seem counterintuitive, it can be readily explained by the use of the loop ordering optimization, which allows to perform multiple time iterations without the need to pull data from DRAM.

Figure 5 shows the predicted performance breakdown into simulation kernels as well as hardware features for all *in silico* models, assuming that the dataset fits in different levels of the memory hierarchy. When data is in the highest level of the cache hierarchy (L1), the most important kernels for all models are state update kernels, and the most relevant hardware feature is the CPU throughput. Additionally, in the G-based models the computation of the exponential (for updating the synaptic states) constitutes a significant portion of the overall execution time. As the dataset increases in size and is only able to fit in lower levels of the cache (L2 or L3) the predicted performance of the G-based models remains

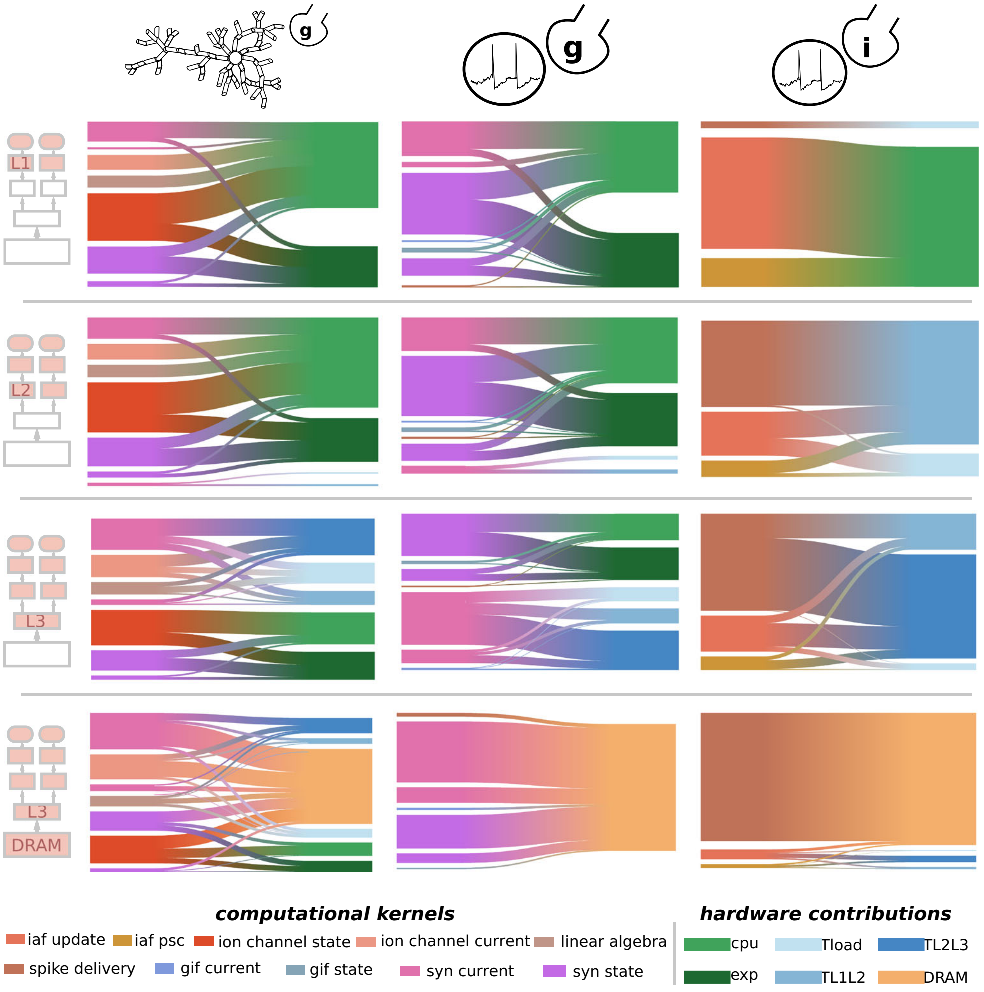

Fig. 5 Predicted shared-memory runtime contributions from computational kernels and hardware features. We assume a single node of the SKX AVX512 and using the maximum number of threads (18 threads). We do not make the assumption of memory bandwidth saturation, but we assume that the loop ordering optimization is used. For each level of the cache hierarchy, we show the breakdown of the total runtime into computational kernels on the left of each box. Furthermore, we show the breakdown of the runtime, as well as the breakdown of individual computational kernels, into hardware contributions on the right of each box. Hardware contributions labels have the following meaning: CPU stands for the execution of non-memory access instructions in the core (excluding the exponential function), exp for the computation of exponential function, T_{load} for the execution of memory access instructions in the core, and the rest for the data traffic time of the relevant datapath

quite stable while that of the Brunel model degrades rapidly, although admittedly our model for the spike delivery kernel in caches might be highly optimistic. In practice, this could be an indication that the Brunel model is bounded by the data path while the G-based models are bounded by the maximum achievable flop rate. Our breakdown analysis confirms this, although for the reference architecture G-based models are best represented by a mix of core-bound and data-bound kernels, especially when the dataset fits only in the L3 cache. Complementarily, in G-based models the relative importance of the core-bound state update kernels gradually loses weight in favour of data-bound current kernels, while in the Brunel model the weight of the spike delivery kernel gradually increases, eventually becoming the most relevant kernel in the execution, as data moves further away from the CPU. In spite of this technique, both point neuron models are clearly dominated by the saturation of the memory bandwidth. In particular, the fact that memory bandwidth is the only factor in determining the performance of the Simplified model can be directly related to the fact that its coupling ratio has a value of 1, as shown in Fig. 2. The performance profile of the Reconstructed model is more diverse, and while 60% of the execution time is still dominated by memory bandwidth, the data transfers between the caches, arithmetic instructions, and throughput of exponential function evaluations also take up a significant portion of the runtime.

Distributed Simulations

An effective strategy for improving simulation performance or to handle larger networks is to dedicate more hardware to the task, distributing the simulated neural network across multiple compute nodes. Here we consider two scenarios, based on the terminology introduced by Singh et al. (1993): a memory constrained scenario and a constant problem size scenario. In the memory constrained scenario the number of neurons is scaled proportionally to the available parallelism (i.e. the number of distributed ranks), while in the constant problem size scenario the number of neurons to be simulated is kept fixed. These scenarios translate respectively to the concepts of weak and strong scaling in high performance computing.

Performance Predictions of Distributed Brain Tissue Simulations We predict the performance of *in silico* models in both scaling scenarios using our performance model, and present the results in Fig. 6. In the memory constrained scenario

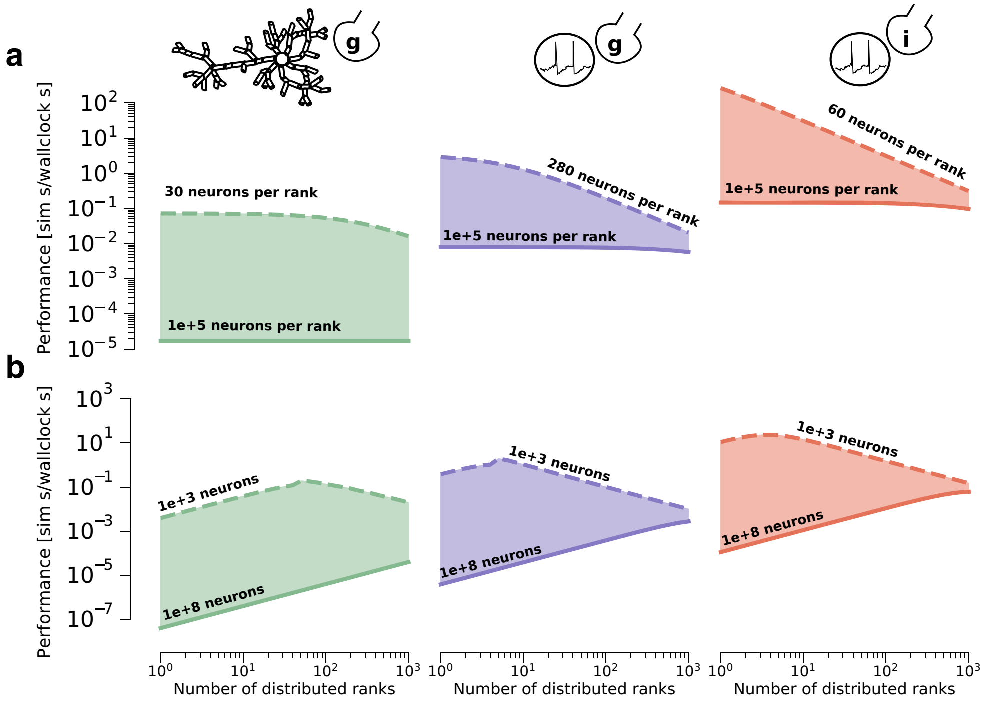

Fig. 6 Performance of distributed scaling and most relevant hardware bottlenecks. The SKX AVX512 architecture with HPE Infiniband EDR is used as reference. **a** Predicted performance of the three *in silico models* in a memory constrained scenario. We consider different numbers of neurons per distributed ranks. The solid lines represent simulations with 10^5 neurons per rank, while the dashed lines represent the estimated minimum number of neurons that is still larger than an L3 cache. The unit of performance is simulated seconds per wallclock second to simulate the whole network. **b** Predicted performance of the three *in silico models* in a constant problem size scenario. We consider different total network sizes. The dashed and solid lines represent simulations with networks of 10^3 and 10^8 neurons respectively. The unit of performance is simulated seconds per wallclock second to simulate the whole network

the simulation performance remains constant regardless of the total number of neurons as long as the number of neurons per rank is sufficiently large. In addition, as expected, the Brunel model has the best predicted performance, beating by up to a factor 10 the performance of the Simplified model and up to a factor 10^4 the performance of the Reconstructed model. For small values of the number of neurons per rank, the performance of the Brunel model degrades with the amount of parallelism, while that of the Reconstructed model is roughly constant, exhibiting only a small degradation at large cluster sizes. In the constant problem size scenario, for all *in silico* models, as long as the network size is sufficiently large, the performance initially improves as we distribute the problem over increasingly more ranks. However, for all *in silico* models there exists a threshold number of ranks after which the benefits from adding hardware become less prominent. Interestingly the striking differences in performance between *in silico* models at small cluster sizes can be evened out quite significantly at large cluster sizes. For example, simulating a large Brunel network on 10 distributed ranks can be roughly four orders of magnitude faster than a Reconstructed network on the same hardware, but the difference between models goes down to two orders of magnitude at large cluster sizes. Scaling to larger cluster sizes after this threshold can be counter-productive, and even result in performance degradation.

Network Latency and Memory Bandwidth are the Main Bottlenecks in Strong Scaling Through introspection of the performance model, we are able to provide an explanation for the observed performance patterns. For each *in silico* model, we investigate the reasons for performance degradation by plotting the most significant hardware bottlenecks for all combinations of network size and cluster size in Fig. 7. We assume that the loop ordering optimization is being used. Even though we do not make the explicit assumption of memory bandwidth saturation, this hardware feature is still among the most relevant for all *in silico* models. Moreover, network bandwidth is never the dominating bottleneck for all *in silico* models and all configurations, while network latency always becomes the most important bottleneck at large cluster sizes. When network latency is the bottleneck we observe a corresponding degradation in performance as more parallel ranks are utilized. This can be directly attributed to the fact that network latency introduces a performance overhead that increases linearly with the number of distributed ranks. Thus we conclude that large-scale simulations are dominated by the latency of the collective communication, and that investigating spike communication strategies such as neighbourhood collectives (Jordan et al. 2018), non-blocking point-to-point schemes (Ananthanarayanan and Modha 2007), asynchronous execution (Magalhaes et al. 2019b) or custom hardware (Navaridas et al. 2012) will be essential to reach brain-scale simulations.

Strawman Architectures and Hardware Design Decisions

A useful feature of our performance model is that it can be generalized to other architectures in a strawman fashion. In this section we focus on providing an educated guess on the performance profile of brain tissue simulations on hardware architectures made with fundamentally different design choices. For this goal we develop the concept of

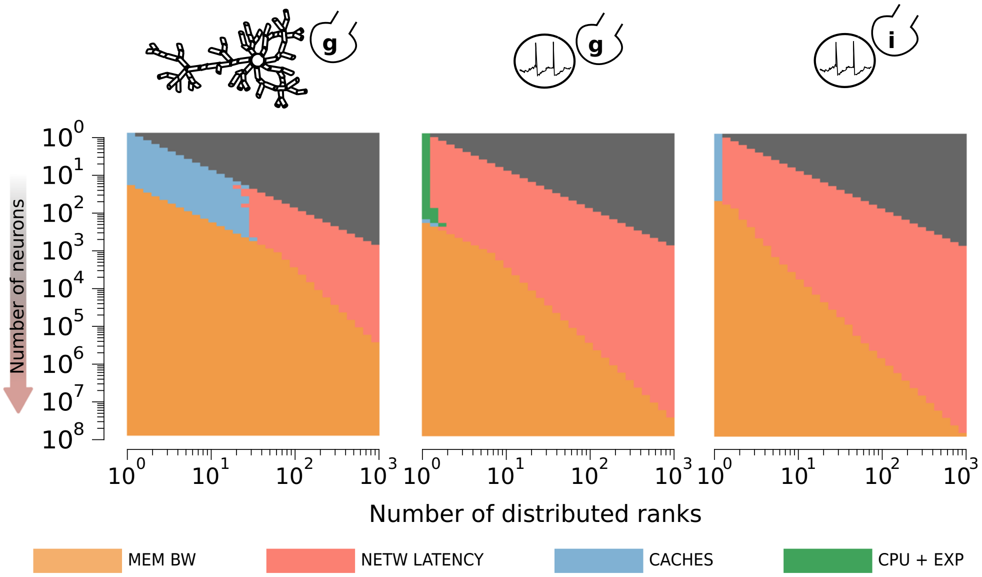

Fig. 7 The SKX AVX512 architecture with HPE Infiniband EDR is used as reference. Most prominent hardware bottlenecks as a function of the total number of neurons (inverted *y* axis) and the number of distributed ranks (*x* axis) in the simulation. The grey areas denote a configuration that would require splitting of individual neurons, and are thus deemed outside the scope of this investigation.

strawman architectures, i.e. models of the hardware that do not necessarily reflect reality perfectly in every detail, but that capture the most salient hardware properties and can be used to explore the design space easily.

Our strawman analysis predicts a speedup of 3x-5x when using the KNL manycore architecture and a speedup of 7x-9x when using the GPU-like architecture compared to the runtime on the reference SKX AVX512 architecture, for all models (Fig. 8). These results have been qualitatively confirmed in cross-platform performance studies (Knight and Nowotny 2018; Kumbhar et al. 2019a; Akar et al. 2019). In this work, however, we are able to dig deeper and identify substantial differences in the hardware bottleneck profiles of individual *in silico* models. The Reconstructed model, for example, is bounded by a mix of scalable features (e.g. arithmetic operations and exponentials) and non-scalable features such as the memory bandwidth. In particular, in architectures with a low clock frequency (KNL) or smaller vector registers (AMD Naples) the non-scalable components constitute more than half of the predicted execution time, indicating that more parallelism could potentially still be beneficial. On the other hand, the Simplified model is predicted to be bounded by memory bandwidth on all architectures. While improving the performance of such a model may seem a difficult task, our previous analysis has shown that improving the coupling ratio of the Simplified model would prove quite beneficial. Finally, The Brunel model is bounded mainly by memory bandwidth on the multicore server-like architectures, but appears bounded by memory level parallelism and latency on the manycore KNL and the GPU architecture. While a more detailed analysis based on actual memory latency values from each architecture would be required to confirm this, it points to an interesting tradeoff between the memory bandwidth and the available parallelism, and indicates that the Brunel model could potentially benefit from high amounts of shared-memory parallelism even with a relatively slower memory bandwidth.

Effect of Model Parameters

Parameters of the *in silico* models have an important, yet often difficult to explain, impact on performance. Firing frequency is commonly cited as one of the most impactful parameters on simulation performance (Yavuz et al. 2016). Firing frequency affects communication by changing the size of the spike message as well as computation by changing the amount of events that must be integrated by neurons. By predicting the performance profile for different values of the firing frequency our analysis shows that, in the median case, the performance of the Reconstructed model is largely unaffected by this parameter, while in the case of the Simplified and Brunel model performance scales linearly with the firing frequency in simulations with large amounts of synaptic activity, as shown in the Supplementary Fig. S13. We further investigate how firing frequency affects the relative importance of hardware bottlenecks, and show in Fig. 9a that while the Reconstructed and Simplified model are largely unaffected by this parameter, the Brunel model's performance profile becomes highly

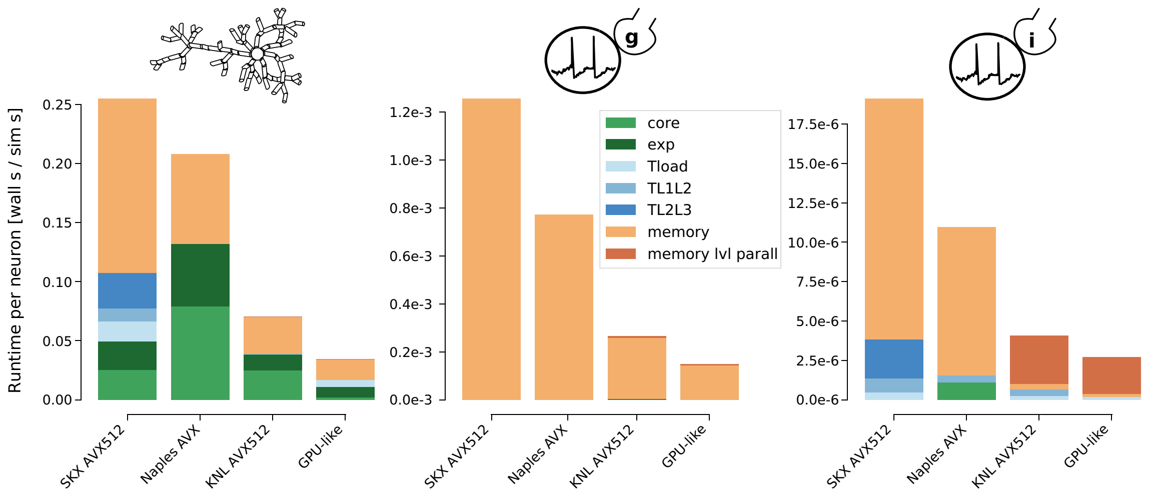

Fig. 8 Breakdown of contributions to total runtime from individual hardware features. Bars represent the total predicted runtime on different strawman hardware architectures for the Reconstructed, Simplified and Brunel model on the left, center and right respectively. Each strawman architecture represents a simplified version of the target hardware, capturing salient hardware properties such as the amount of available shared-memory parallelism, the memory bandwidth, the clock frequency and the memory hierarchy. The rest of the hardware details, most notably the throughput of instructions in the core and the memory level parallelism and latency, were obtained by adapting the corresponding known values from the reference Skylake architecture. The meaning of individual hardware contributions is based on the corresponding ECM model dimension as in Fig. 5

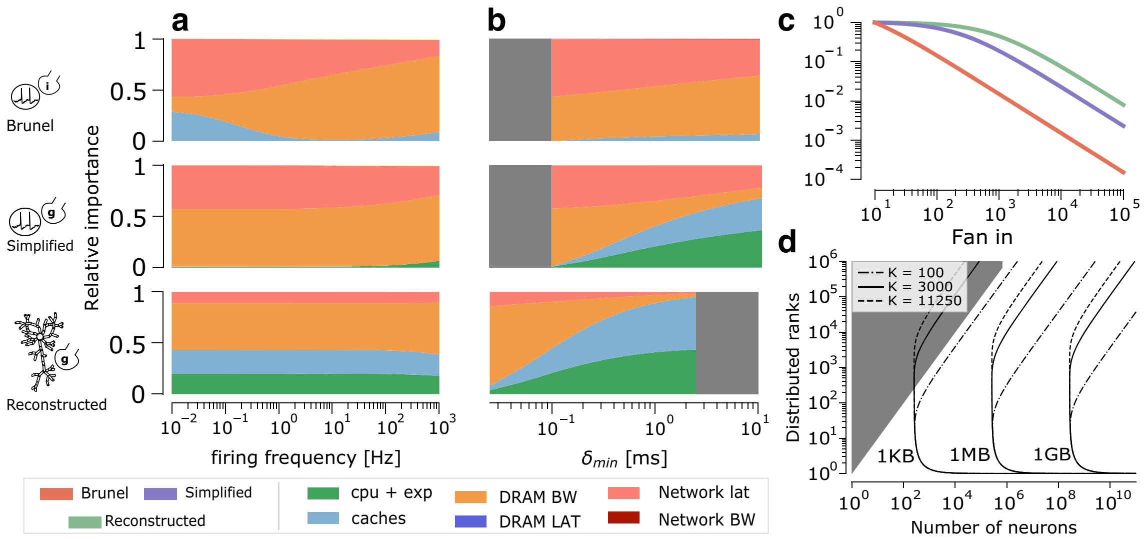

Fig. 9 **a** Stacked plot of the mean relative contributions from hardware features as a function of the average firing frequency of neurons in the simulation. The mean was extracted by simulating 1000 randomly generated simulation configurations, defined by number of neurons and number of distributed ranks. **b** Stacked plot of the mean relative contributions from hardware features as a function of the δ_{min}. The range of acceptable values for δ_{min} changes across different *in silico* models because they were computed as multiples of the model's timestep, hence the greyed-out areas. **c** Number of neurons able to fit in 1 GB of memory, normalized by the memory requirements of a model with 10 incoming synapses, as a function of the average fan in per neuron. **d** Contour plot of predicted memory requirements of the connections table, as a function of the total number of neurons (x axis) and the number of distributed ranks (y axis). The contour levels corresponding to 1 KB, 1 MB and 1 GB are shown for different values of the fan in

skewed towards the memory bandwidth at large values of the firing frequency. Interestingly, our analysis also shows that the minimum network delay also has a significant impact on performance, despite the fact that it does not change the total number of operations to be performed. In particular, for point neuron models there is a transition from a regime dominated by communication to one dominated by computation, while the Reconstructed model is dominated by computation for all values of the minimum network delay, as shown in Fig. 9b. Finally, we show how the average number of incoming synaptic connections presents a subtle tradeoff of increasing the computational load (thus decreasing performance) while at the same time increasing memory requirements, and thus potentially the amount of distributed parallelism required to simulate a given number of neurons (thus increasing performance). We examine two ways in which fan influences memory requirements: Fig. 9c shows the fraction of neurons that can fit in 1 GB of memory for different values of the fan in, solely by virtue of the additional parameters and state variables required to represent the corresponding synapses; Fig. 9d further investigates the memory requirements of the connection table, required by each distributed rank to determine whether a specific source neuron has any local postsynaptic connections on that rank. We refer the interested reader to the Supplementary Material S2 for additional details in the analysis.

Synaptic Plasticity The scope of this work is limited to the investigation of the performance properties of the inference phase of biological neural networks, assuming that the learning process, if any, is carried out in a separate experiment. However, using insight from our analysis, we can still show that synaptic plasticity would not likely change the performance profiles that we have identified in a dramatic way. First of all, given the event-driven and unpredictable nature of synaptic activation, it should be noted that plasticity kernels share the same memory access pattern as the spike delivery kernels, and are thus also potentially affected by memory level parallelism and latency. While long-term plasticity rules typically also involve computationally heavy operations, it is reasonable to assume that the memory level parallelism will be the dominant hardware bottleneck, given that the G-based spike delivery kernel has a similar profile. Therefore, we expect that in the Reconstructed and Simplified models the impact of synaptic plasticity rules would be negligible, since the event-driven portion of the runtime was already found to be small, and although synaptic plasticity would definitely increase it, it is unlikely that this would ultimately amount to a significant effect. In the Brunel model, which is dominated by event-driven computations, synaptic plasticity would simply exacerbate this profile as it shares the same bottleneck profile as the spike delivery kernel.

Discussion

In this work we have delivered a quantitative characterization of the performance properties of different published *in silico* models at the core of state-of-the-art brain tissue simulations. Using a grey-box model that combines biological and algorithmic properties with hardware specifications we have identified performance bottlenecks under different simulation regimes, corresponding to a variety of prototypical scientific questions that can be answered by simulations of biological neural networks.

General Purpose Computing has Sustained a Diverse Performance Landscape up to Now Our results show that there exists a large diversity of performance profiles and bottlenecks that shape the landscape of brain tissue simulations, corresponding to the diversity of sizes and scales at which research questions in simulation neuroscience can be asked. Thus, our research highlights that the computational neuroscience community is currently greatly benefitting from the adaptability of general purpose computing, exploiting the ease of development and high performance capability to explore different areas of the modeling landscape.

Memory Bandwidth and Network Latency Severely Limit Maximum Filling and Real Time Strong Scaling Using a state-of-the-art HPC server CPU and cluster as a reference, our analysis revealed that all the *in silico* models saturate the memory bandwidth using quite a small number of shared memory threads. Even when algorithmic improvements are put into place to mitigate this effect we have identified that the coupling ratio, a dimensionless number that counts the number of timesteps in a mininum network delay period, strongly regulates the saturation of memory bandwidth and, in the extreme case of the Simplified model analyzed here, effectively prevents any benefit to be gained from the effort of developing a more efficient algorithm. Additionally, we discovered that it is not the level of morphological detail, but rather the formalism used to represent synapses, that is the most important factor in explaining the memory bandwidth saturation profile, with G-based models saturating much faster than the I-based model. Our analysis of strawman architectures has shown that the extremely fast memory bandwidth of current manycore architectures is currently able to sustain large amounts of shared-memory parallelism for all models, with performance improvements proportional to bandwidth improvements. In distributed simulations we identified the network latency, and not the network bandwidth, as the major bottleneck for scaling to very large networks or very large cluster sizes. This provides a new motivation and justification for the extensive efforts described in Navaridas et al. (2012) in designing a specific communication infrastructure for the SpiNNaker neuromorphic system.

Model-Specific Features have a Significant Impact on Shared-Memory Performance Inspection of our performance model allowed us to pinpoint which kernels, hardware specifications and model parameters have the largest impact on performance. The Brunel model based on the I-based formalism and IAF neurons is mainly bounded by the spike delivery kernel, which exhibits a good shared-memory scaling behaviour and, in the case of extreme strong scaling, a strong dependence on the inter-cache data paths for good performance. The two G-based models we analyzed, i.e. Simplified and Reconstructed, have a similar shared-memory scaling behaviour, mainly driven by the *current* kernels required to compute the contributions of individual synapses (and ion channels) to the membrane potential equation. However, while the Simplified model is 100% dominated by memory bandwidth, the morphologically detailed Reconstructed model is dominated partially (around 40%) by other hardware components such as caches and CPU throughput. Using strawman architectures, we have confirmed that these profiles are valid for a wide range of hardware design choices, with differences arising only when memory bandwidth is not fully saturated. It becomes clear that a performance model and a detailed performance analysis are fundamental tools to disentagle the complex web of relationships between *in silico* models, their software implementation and hardware concretization.

Static and Dynamic Model Parameters Affect Performance in Significant but Subtle Ways Finally, we examined the impact of model parameters on the performance profiles described above. We found that firing frequency, but surprisingly also minimum network delay, can have a large impact on determining which hardware features may constitute a performance bottleneck. For firing frequency it is obvious that larger values correspond to more operations required by the simulation algorithm, and thus a lower performance, but our analysis shows that different values of the firing frequency also change the relative importance of hardware features. Interestingly we found that the minimum network delay, in spite of it not affecting the total number of operations per simulated second, can have an effect on performance simply by shifting the importance of the hardware bottlenecks. We also found that the average number of incoming connections per neuron plays a subtle role in influencing performance. Trivially, a larger fan in increases the computational requirements of a single neuron. However, it also increases the memory capacity requirements, thus requring a larger degree of parallelism to handle the same network size. This creates a tradeoff between performance degradation arising from larger computational requirements and performance improvement from parallelism requirements.

Limits and Future Improvements In this work we have concentrated solely on the aspect of maximizing performance, without considering limitations such as cost or energy. However, it must be stated that energy efficiency is a central issue in the computational neuroscience community, and one of the main selling points of neuromorphic hardware (Cassidy et al. 2014; Stromatias et al. 2013). Therefore, a meaningful extension to this work would be to incorporate a model for power consumption alongside performance prediction, as a way to constrain the feasibility and efficiency of certain simulation configurations. To achieve this, one could exploit already established power consumption models that are easily integrated with the ECM and have been shown to provide valuable insight into the power and performance properties of simulation kernels (Hager et al. 2016; Hofmann et al. 2018). Moreover, our analysis has been focused on computational and communication kernels, effectively neglecting other aspects of simulation performance such as generation of stimuli, random number generation and managing the queue of synaptic events. While these kernels constitute necessary steps in the simulation of brain tissue, the goal of our investigation is to study the performance properties related to the mathematical modeling of neurons, and not implementation and hardware details such as the most efficient random number generation strategy. Despite excluding some parts of the simulation algorithm, our analysis still maintains a lot of relevance with regards to the overall simulation performance. Indeed, in both G-based and I-based models, it was found that computational and communication kernels often constitute more than 90% of the total simulation runtimes (Kumbhar et al. 2019a; Ewart et al. 2015; Schenck et al. 2014; Peyser and Schenck 2015). From the modeling point of view, an important aspect that we have neglected in this analysis is synaptic plasticity. A large portion of research questions that require brain tissue simulations involve learning and synaptic plasticity, so this represents an important extension to our analysis. However, in this work we decided to concentrate on the inference part of brain tissue simulations because the diversity and complexity of plasticity models warrants a separate analysis. However, we have shown that we do not expect synaptic plasticity to significantly change our analysis. In the future, given that the performance modeling infrastructure is already in place, we believe that the addition of plasticity for a more detailed analysis would not be a technical challenge, although it would considerably complexify the resulting analysis. Even though we already considered potential hardware improvements in our analysis, it would be interesting to extend this study to include hardware with different a design space such as the non-overlapping caches of AMD CPUs (Hager 2017) or massive SIMD parallelism of GPGPUs (Knight and Nowotny 2018). Finally, the methods developed in this paper can be extended to different simulation approaches ranging from different communication strategies (Kozloski and Wagner 2011; Ananthanarayanan and Modha 2007) up to fully asynchronous executions (Magalhães and Schürmann 2019).

Performance Modeling is Required to Enable Next-Generation High Performance Brain Tissue Simulations Our analysis shows that, if future iterations of general-purpose hardware architectures maintain the same balance as the current state-of-the-art, it will be very difficult to achieve fast, large scale simulations of brain tissue. Even if hardware peak performance were to improve significantly over the next years, the required speedup could only be achieved via specifically targeted advancements and under very restrictive simulation and model configurations. To support the next generation of brain tissue simulations, the community must therefore focus on the design of dedicated hardware.

In this work we have shown that different modeling choices and abstractions lead to a large diversity in the performance landscape of cellular-level simulations, in spite of being representations of the same biological phenomenon, thus making the task of designing the hardware and software necessary to achieve high performance simulations extremely difficult. The level of diversity means that design tradeoffs will inevitably require very restrictive decisions about the scale, model and configuration of the target simulation, such as e.g. (Navaridas et al. 2012; Knight and Nowotny 2018). For example, in the context of accelerating small networks of neurons up to real-time, we predict that I-based models would benefit from architectures with high memory level parallelism and a simple cache hierarchy, while for G-based models the cache bandwidth as well as the throughput of CPU arithmetic operations must be improved concurrently in order observe the desired speedup. Additionally, simulation parameters can influence the performance profile in a tangible way. In the case of the Simplified model, for example, simulations are on average bounded by memory bandwidth and network latency at the current value of the minimum network delay, but the profile would change drastically if the minimum network delay were increased to the same value of the Brunel model, with memory bandwidth losing importance in favour of cache and CPU throughput. Even in ideal cases where the *in silico* experiment falls perfectly within the design space of the hardware and software being used, simulation dynamics outside of the control of the modeler such as firing frequency can rapidly push the performance of simulations towards suboptimal regimes.

Conclusions

We have demonstrated the diversity and complexity of the performance landscape of brain tissue simulations, using an analytical performance model that combines algorithm definitions and hardware specifications as an exploration tool. Thanks to the analytical modeling tools developed by us, we are able to gain insight on intrinsic model properties at the root of the observed performance variability across models, and to pinpoint hardware and software culprits for performance bottlenecks in brain tissue simulations. We conclude that, even though state-of-the-art published simulations are currently greatly benefitting from general purpose computing, it is likely that reaching fast, brain-scale simulations on general purpose HPC platforms will not be feasible in the near future. In order to achieve this goal, computational neuroscience modelers must cooperate with software developers and hardware designers. Ultimately, we stress that performance modeling represents a powerful tool to enable communication between these communities We believe that our work embodies a concrete step in defining and understanding key performance properties of a wide variety of *in silico* models, necessary to enable the next generation of brain tissue simulations.

Resource Sharing Statement

All the information about the *in silico* models was obtained directly from the corresponding publications and is publicly available.

Acknowledgements The authors gratefully acknowledge the compute resources and support provided by the Erlangen Regional Computing Center (RRZE), and in particular Georg Hager for the fruitful discussions regarding the ECM model and the interpretation of performance predictions. We are also indebted to the BlueBrain HPC team, and in particular Bruno Magalhaes and Pramod Kumbhar, for helpful support and discussion regarding CoreNEURON.

References

Aamir, S.A., Stradmann, Y., Müller, P., Pehle, C., Hartel, A., Grübl, A., Schemmel, J., Meier, K. (2018). An accelerated lif neuronal network array for a large-scale mixed-signal neuromorphic architecture. IEEE Transactions on Circuits and Systems I: Regular Papers (99), 1–14. https://doi.org/10.1109/tcsi.2018.2840718.

Akar, N.A., Cumming, B., Karakasis, V., Küsters, A., Klijn, W., Peyser, A., Yates, S. (2019). Arbor: a morphologically-detailed neural network simulation library for contemporary high-performance computing architectures. In *2019 27th euromicro international conference on parallel, distributed and network-based processing (PDP)* (pp. 274-282): IEEE, https://doi.org/10.1109/empdp.2019.8671560.

Alexandrov, A., Ionescu, M.F., Schauser, K.E., Scheiman, C. (1997). Loggp: Incorporating long messages into the logp model for parallel computation. *Journal of Parallel and Distributed Computing*, *44*(1), 71–79. https://doi.org/10.1006/jpdc.1997.1346.

Ananthanarayanan, R., & Modha, D.S. (2007). Anatomy of a cortical simulator. In *Proceedings of the 2007 ACM/IEEE conference on Supercomputing* (p. 3): ACM, https://doi.org/10.1145/1362622.1362627.

Ananthanarayanan, R., Esser, S.K., Simon, H.D., Modha, D.S. (2009). The cat is out of the bag: cortical simulations with 10 9 neurons, 10 13 synapses. In *Proceedings of the conference on high performance computing networking, storage and analysis* (p. 63): ACM.

Asanovic, K., Bodik, R., Demmel, J., Keaveny, T., Keutzer, K., Kubiatowicz, J., Morgan, N., Patterson, D., Sen, K., Wawrzynek, J., et al. (2009). A view of the parallel computing landscape. *Communications of the ACM*, *52*(10), 56–67. https://doi.org/10.1145/1562764.1562783.

Benjamin, B.V., Gao, P., McQuinn, E., Choudhary, S., Chandrasekaran, A.R., Bussat, J.M., Alvarez-Icaza, R., Arthur, J.V., Merolla, P.A., Boahen, K. (2014). Neurogrid: a mixed-analog-digital multichip system for large-scale neural simulations. *Proceedings of the IEEE*, *102*(5), 699–716. https://doi.org/10.1109/jproc.2014.2313565.

Brette, R., & Goodman, D.F. (2011). Vectorized algorithms for spiking neural network simulation. *Neural Computation*, *23*(6), 1503–1535. https://doi.org/10.1162/neco_a_00123.

Brette, R., & Goodman, D.F. (2012). Simulating spiking neural networks on gpu. *Network: Computation in Neural Systems*, *23*(4), 167–182. https://doi.org/10.3109/0954898x.2012.730170.

Brette, R., Rudolph, M., Carnevale, T., Hines, M., Beeman, D., Bower, J.M., Diesmann, M., Morrison, A., Goodman, P.H., Harris, J.r. F.C., et al. (2007). Simulation of networks of spiking neurons: a review of tools and strategies. *Journal of Computational Neuroscience*, *23*(3), 349–398. https://doi.org/10.1007/s10827-007-0038-6.

Brunel, N. (2000). Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons. *Journal of Computational Neuroscience*, *8*(3), 183–208. https://doi.org/10.1016/s0925-2312(00)00179-x.

Calotoiu, A., Hoefler, T., Poke, M., Wolf, F. (2013). Using automated performance modeling to find scalability bugs in complex codes. In *Proceedings of the ACM/IEEE Conference on Supercomputing (SC13), Denver, CO, USA* (pp. 1-12): ACM, https://doi.org/10.1145/2503210.2503277.

Cassidy, A.S., Alvarez-Icaza, R., Akopyan, F., Sawada, J., Arthur, J.V., Merolla, P.A., Datta, P., Tallada, M.G., Taba, B., Andreopoulos, A., et al. (2014). Real-time scalable cortical computing at 46 giga-synaptic ops/watt with. In *Proceedings of the international conference for high performance computing, networking, storage and analysis* (pp. 27–38): IEEE Press.

Cremonesi, F., Hager, G., Wellein, G., Schürmann, F. (2019). Analytic performance modeling and analysis of detailed neuron simulations. The International Journal of High Performance Computing Applications. In review.

Culler, D., Karp, R., Patterson, D., Sahay, A., Schauser, K.E., Santos, E., Subramonian, R., Von Eicken, T. (1993). Logp: Towards a realistic model of parallel computation. In *ACM Sigplan notices*, (Vol. 28 pp. 1-12): ACM, https://doi.org/10.1145/155332.155333.

Dennard, R.H., Gaensslen, F.H., Rideout, V.L., Bassous, E., LeBlanc, A.R. (1974). Design of ion-implanted mosfet's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, *9*(5), 256–268. https://doi.org/10.1109/jproc.1999.752522.

Eichner, H., Klug, T., Borst, A. (2009). Neural simulations on multi-core architectures. *Frontiers in Neuroinformatics*, *3*, 21. https://doi.org/10.3389/neuro.11.021.2009.

Ewart, T., Yates, S., Cremonesi, F., Kumbhar, P., Schürmann, F., Delalondre, F. (2015). Performance evaluation of the IBM POWER8 architecture to support computational neuroscientific application using morphologically detailed neurons. In: Proc. 6th int. workshop on perfomance modeling, benchmarking, and simulation of high performance computing systems, ACM.

Fidjeland, A.K., Roesch, E.B., Shanahan, M.P., Luk, W. (2009). Nemo: a platform for neural modelling of spiking neurons using gpus. In *2009 20th IEEE international conference on application-specific systems, architectures and processors* (pp. 137–144): IEEE, https://doi.org/10.1109/asap.2009.24.

Gerstner, W., Kistler, W.M., Naud, R., Paninski, L. (2014). *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge: Cambridge University Press.

Hagen, E., Dahmen, D., Stavrinou, M.L., Lindén, H., Tetzlaff, T., van Albada, S.J., Grün, S., Diesmann, M., Einevoll, G.T. (2016). Hybrid scheme for modeling local field potentials from point-neuron networks. *Cerebral Cortex*, 1–36. https://doi.org/10.1186/1471-2202-16-s1-p67.

Hagen, E., Næss, S., Ness, T.V., Einevoll, G.T. (2018). Multimodal modeling of neural network activity: Computing lfp, ecog, eeg, and meg signals with lfpy 2.0. *Frontiers in Neuroinformatics*, 12. https://doi.org/10.3389/fninf.2018.00092.

Hager, G. (2017). Benchmarking the memory hierarchy of the new amd ryzen cpu using the vector triad. Georg Hager's blog https://blogs.fau.de/hager/archives/7810.

Hager, G., Treibig, J., Habich, J., Wellein, G. (2016). Exploring performance and power properties of modern multi-core chips via simple machine models. *Concurrency and Computation: Practice and Experience*, *28*(2), 189–210. https://doi.org/10.1002/cpe.3180.

Hager, G., Eitzinger, J., Hornich, J., Cremonesi, F., Alappat, C.L., Röhl, T., Wellein, G. (2018). Applying the execution-cache-memory model: Current state of practice, poster at Supercomputing 2018.

Hardavellas, N., Ferdman, M., Falsafi, B., Ailamaki, A. (2011). Toward dark silicon in servers. *IEEE Micro*, *31*(4), 6–15. https://doi.org/10.1109/mm.2011.77.

Helias, M., Kunkel, S., Masumoto, G., Igarashi, J., Eppler, J.M., Ishii, S., Fukai, T., Morrison, A., Diesmann, M. (2012). Supercomputers ready for use as discovery machines for neuroscience. *Frontiers in Neuroinformatics*, *6*, 26. https://doi.org/10.3389/fninf.2012.00026.

Hines, M., Kumar, S., Schürmann, F. (2011). Comparison of neuronal spike exchange methods on a blue gene/p supercomputer. *Frontiers in Computational Neuroscience*, *5*, 49. https://doi.org/10.3389/fncom.2011.00049.

Hines, M.L., Markram, H., Schürmann, F. (2008). Fully implicit parallel simulation of single neurons. *Journal of Computational Neuroscience*, *25*(3), 439–448. https://doi.org/10.1186/1471-2202-8-s2-p6.

Hoefler, T., Schneider, T., Lumsdaine, A. (2009). Loggp in theory and practice–an in-depth analysis of modern interconnection networks and benchmarking methods for collective operations. *Simulation Modelling Practice and Theory*, *17*(9), 1511–1521. https://doi.org/10.1016/j.simpat.2009.06.007.

Hofmann, J., Eitzinger, J., Fey, D. (2015). Execution-cache-memory performance model: Introduction and validation. arXiv:150903118.

Hofmann, J., Hager, G., Wellein, G., Fey, D. (2017). An analysis of core-and chip-level architectural features in four generations of intel server processors. In *International supercomputing conference* (pp. 294–314): Springer, https://doi.org/10.1007/978-3-319-58667-0_16.

Hofmann, J., Hager, G., Fey, D. (2018). On the accuracy and usefulness of analytic energy models for contemporary multicore processors. In Yokota, R., Weiland, M., Keyes, D., Trinitis, C. (Eds.) *International conference on high performance computing* (pp. 22–43). Cham: Springer International Publishing, https://doi.org/10.1007/978-3-319-92040-5_2.

Hofmann, J., Alappat, C.L., Hager, G., Fey, D., Wellein, G. (2019). Bridging the architecture gap: Abstracting performance-relevant properties of modern server processors. arXiv:190700048.

Indiveri, G., Linares-Barranco, B., Hamilton, T.J., Van Schaik, A., Etienne-Cummings, R., Delbruck, T., Liu, S.C., Dudek, P., Häfliger, P., Renaud, S., et al. (2011). Neuromorphic silicon neuron circuits. *Frontiers in Neuroscience*, *5*, 73. https://doi.org/10.3389/fnins.2011.00073.

Intel (2017). Intel architecture code analyzer. https://software.intel.com/en-us/articles/intel-architecture-code-analyzer.

Ippen, T., Eppler, J.M., Plesser, H.E., Diesmann, M. (2017). Constructing neuronal network models in massively parallel environments. *Frontiers in Neuroinformatics*, *11*, 30. https://doi.org/10.3389/fninf.2017.00030.

Izhikevich, E.M., & Edelman, G.M. (2008). Large-scale model of mammalian thalamocortical systems. *Proceedings of the National Academy of Sciences*, *105*(9), 3593–3598. https://doi.org/10.1073/pnas.0712231105.

Jeffers, J., Reinders, J., Sodani, A. (2016). *Intel Xeon Phi processor high performance programming: knights landing edition*. Burlington: Morgan Kaufmann.

Jia, Z., Maggioni, M., Staiger, B., Scarpazza, D.P. (2018). Dissecting the nvidia volta gpu architecture via microbenchmarking. arXiv:180406826.

Jordan, J., Ippen, T., Helias, M., Kitayama, I., Sato, M., Igarashi, J., Diesmann, M., Kunkel, S. (2018). Extremely scalable spiking neuronal network simulation code: from laptops to exascale computers. *Frontiers in Neuroinformatics*, *12*, 2. https://doi.org/10.3389/fninf.2018.00002.

Knight, J.C., & Nowotny, T. (2018). Gpus outperform current hpc and neuromorphic solutions in terms of speed and energy when simulating a highly-connected cortical model. *Frontiers in Neuroscience*, *12*, 941. https://doi.org/10.3389/fnins.2018.00941.

Kozloski, J., & Wagner, J. (2011). An ultrascalable solution to large-scale neural tissue simulation. *Frontiers in Neuroinformatics*, *5*, 15. https://doi.org/10.3389/fninf.2011.00015.

Kumbhar, P., Hines, M., Ovcharenko, A., Mallon, D.A., King, J., Sainz, F., Schürmann, F., Delalondre, F. (2016). Leveraging a cluster-booster architecture for brain-scale simulations. In *International conference on high performance computing* (pp. 363–380): Springer, https://doi.org/10.1007/978-3-319-41321-1_19.

Kumbhar, P.S., Sivagnanam, S., Yoshimoto, K., Hines, M., Carnevale, T., Majumdar, A. (2018). Performance analysis of computational neuroscience software neuron on knights corner many core processors. In *Workshop on software challenges to exascale computing* (pp. 67–76): Springer, https://doi.org/10.1007/978-981-13-7729-7_5.

Kumbhar, P., Awile, O., Keegan, L., Alonso, J.B., King, J., Hines, M., Schürmann, F. (2019a). An optimizing multi-platform source-to-source compiler framework for the neuron modeling language. arXiv:190502241.

Kumbhar, P., Hines, M., Fouriaux, J., Ovcharenko, A., King, J., Delalondre, F., Schürmann, F. (2019b). Coreneuron: an optimized compute engine for the neuron simulator. Frontiers in Neuroinformatics. https://doi.org/10.3389/fninf.2019.00063.

Kunkel, S., Schmidt, M., Eppler, J.M., Plesser, H.E., Masumoto, G., Igarashi, J., Ishii, S., Fukai, T., Morrison, A., Diesmann, M., et al. (2014). Spiking network simulation code for petascale computers. *Frontiers in Neuroinformatics*, *8*, 78. https://doi.org/10.3389/fninf.2014.00078.

Levinthal, D. (2014). Cycle accounting analysis on intel core 2 processors. Tech. rep., whitepaper.

Lytton, W.W., & Hines, M.L. (2005). Independent variable time-step integration of individual neurons for network simulations. *Neural Computation*, *17*(4), 903–921. https://doi.org/10.1162/0899766053429453.

Magalhães, B., & Schürmann, F. (2019). Fully-asynchronous cache-efficient simulation of detailed neural networks. https://doi.org/10.1007/978-3-030-22744-9_33.

Magalhaes, B., Hines, M., Sterling, T., Schürmann, F. (2019a). Asynchronous branch-parallel simulation of detailed neuron models (under review). Frontiers in Neuroinformatics. https://doi.org/10.3389/fninf.2019.00054.

Magalhaes, B., Hines, M., Sterling, T., Schürmann, F. (2019b). Exploiting Flow graph of system of odes to accelerate the simulation of biologically-detailed neural networks. In *Proceedings of 2019 IEEE international parallel and distributed processing symposium (IPDPS), IEEE.*

Mamadou, H.N., Nanri, T., Murakami, K. (2006). Collective communication costs analysis over gigabit ethernet and infiniband. In *International conference on high-performance computing* (pp. 547–559): Springer, https://doi.org/10.1007/11945918_52.

Markram, H., Muller, E., Ramaswamy, S., Reimann, M.W., Abdellah, M., Sanchez, C.A., Ailamaki, A., Alonso-Nanclares, L., Antille, N., Arsever, S., et al. (2015). Reconstruction and simulation of neocortical microcircuitry. *Cell*, *163*(2), 456–492. https://doi.org/10.1016/j.cell.2015.09.029.

McCalpin, J.D. (1995). Memory bandwidth and machine balance in current high performance computers. *IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter*, 19–25.

Moore, G.E. (1995). Lithography and the future of Moore's law. In *Integrated circuit metrology, inspection, and process control IX, International Society for Optics and Photonics*, (Vol. 2439 pp. 2–18).

Morrison, A., Mehring, C., Geisel, T., Aertsen, A., Diesmann, M. (2005). Advancing the boundaries of high-connectivity network simulation with distributed computing. *Neural Computation*, *17*(8), 1776–1801. https://doi.org/10.1162/0899766054026648.

Navaridas, J., Luj'n, M., Plana, L.A., Miguel-Alonso, J., Furber, S.B. (2012). Analytical assessment of the suitability of multicast communications for the spinnaker neuromimetic system. In *2012 IEEE 14th international conference on high performance computing and communication & 2012 IEEE 9th international conference on embedded software and systems* (pp. 1–8): IEEE, https://doi.org/10.1109/hpcc.2012.11.

Nolte, M., Reimann, M.W., King, J.G., Markram, M., Muller, E.B. (2019). Cortical reliability amid noise and chaos. *Nat Commun*, *10*, 3792. https://doi.org/10.1038/s41467-019-11633-8.

Ovcharenko, A., Kumbhar, P.S., Hines, M.L., Cremonesi, F., Ewart, T., Yates, S., Schürmann, F., Delalondre, F. (2015). Simulating morphologically detailed neuronal networks at extreme scale. In *PARCO* (pp. 787–796).

Painkras, E., Plana, L.A., Garside, J., Temple, S., Galluppi, F., Patterson, C., Lester, D.R., Brown, A.D., Furber, S.B. (2013). Spinnaker: a 1-w 18-core system-on-chip for massively-parallel neural network simulation. *IEEE Journal of Solid-State Circuits*, *48*(8), 1943–1953. https://doi.org/10.1109/jssc.2013.2259038.

Peyser, A., & Schenck, W. (2015). The nest neuronal network simulator: Performance optimization techniques for high performance computing platforms. In *Society for Neuroscience Annual Meeting, Jülich Supercomputing Center, FZJ-2015-06261*.

Potjans, T.C., & Diesmann, M. (2012). The cell-type specific cortical microcircuit: relating structure and activity in a full-scale spiking network model. *Cerebral Cortex*, *24*(3), 785–806. https://doi.org/10.1093/cercor/bhs358.

Pozzorini, C., Mensi, S., Hagens, O., Naud, R., Koch, C., Gerstner, W. (2015). Automated high-throughput characterization of single neurons by means of simplified spiking models. *PLoS Computational Biology*, *11*(6), e1004275. https://doi.org/10.1371/journal.pcbi.1004275.

Rössert, C., Pozzorini, C., Chindemi, G., Davison, A.P., Eroe, C., King, J., Newton, T.H., Nolte, M., Ramaswamy, S., Reimann, M.W., et al. (2016). Automated point-neuron simplification of data-driven microcircuit models. arXiv:160400087.

Schenck, W., Adinetz, A., Zaytsev, Y., Pleiter, D., Morrison, A. (2014). Performance model for large–scale neural simulations with nest. In *Extended poster abstracts of the SC14 conference for supercomputing*.

Simonite, T. (2016). Moore's law is dead. now what. MIT Technology review.

Singh, J.P., Hennessy, J.L., Gupta, A. (1993). Scaling parallel programs for multiprocessors: Methodology and examples. *Computer*, *26*(7), 42–50. https://doi.org/10.1109/MC.1993.274941.

Stengel, H., Treibig, J., Hager, G., Wellein, G. (2015). Quantifying performance bottlenecks of stencil computations using the Execution-Cache-Memory model. In *Proceedings of the 29th ACM international conference on supercomputing, ACM, New York, NY, USA, ICS '15. https://doi.org/10.1145/2751205.2751240*.

Stromatias, E., Galluppi, F., Patterson, C., Furber, S. (2013). Power analysis of large-scale, real-time neural networks on spinnaker. In *The 2013 international joint conference on neural networks (IJCNN)* (pp. 1–8): IEEE, https://doi.org/10.1109/ijcnn.2013.6706927.

Thakur, R., Rabenseifner, R., Gropp, W. (2005). Optimization of collective communication operations in mpich. *The International Journal of High Performance Computing Applications*, *19*(1), 49–66. https://doi.org/10.1177/1094342005051521.

Treibig, J., & Hager, G. (2010). Introducing a performance model for bandwidth-limited loop kernels. In *Parallel processing and applied mathematics* (pp. 615–624): Springer, https://doi.org/10.1007/978-3-642-14390-8_64.

Valiant, L.G. (1990). A bridging model for parallel computation. *Communications of the ACM*, *33*(8), 103–111. https://doi.org/10.1145/79173.79181.

van Albada, S.J., Rowley, A.G., Senk, J., Hopkins, M., Schmidt, M., Stokes, A.B., Lester, D.R., Diesmann, M., Furber, S.B. (2018). Performance comparison of the digital neuromorphic hardware spinnaker and the neural network simulation software nest for a full-scale cortical microcircuit model. *Frontiers in Neuroscience*, *12*, 291. https://doi.org/10.3389/fnins.2018.00291.

Williams, S., Waterman, A., Patterson, D. (2009). Roofline: an insightful visual performance model for multicore architectures. *Commun ACM*, *52*(4), 65–76. http://doi.acm.org/10.1145/1498765.1498785.

Wunderlich, T., Kungl, A.F., Hartel, A., Stradmann, Y., Aamir, S.A., Grübl, A., Heimbrecht, A., Schreiber, K., Stöckel, D., Pehle, C., et al. (2018). Demonstrating advantages of neuromorphic computation: a pilot study. arXiv:181103618. https://doi.org/10.3389/fnins.2019.00260.

Yavuz, E., Turner, J., Nowotny, T. (2016). Genn: a code generation framework for accelerated brain simulations. *Scientific Reports*, *6*, 18854. https://doi.org/10.1038/srep18854.

Zenke, F., & Gerstner, W. (2014). Limits to high-speed simulations of spiking neural networks using general-purpose computers. *Frontiers in Neuroinformatics*, *8*, 76. https://doi.org/10.3389/fninf.2014.00076.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.