Compass: A scalable simulator for an architecture for Cognitive Computing

Robert Preissl, Theodore M. Wong, Pallab Datta, Myron Flickner,
Raghavendra Singh, Steven K. Esser, William P. Risk, Horst D. Simon[†], and Dharmendra S. Modha
IBM Research - Almaden, San Jose CA 95120
[†] Lawrence Berkeley National Lab, Berkeley, CA 94720
Contact e-mail: mdflickner@us.ibm.com

***Abstract*—Inspired by the function, power, and volume of the organic brain, we are developing TrueNorth, a novel modular, non-von Neumann, ultra-low power, compact architecture. TrueNorth consists of a scalable network of neurosynaptic cores, with each core containing neurons, dendrites, synapses, and axons. To set sail for TrueNorth, we developed Compass, a multi-threaded, massively parallel functional simulator and a parallel compiler that maps a network of long-distance pathways in the macaque monkey brain to TrueNorth. We demonstrate near-perfect weak scaling on a 16 rack IBM® Blue Gene®/Q (262144 CPUs, 256 TB memory), achieving an unprecedented scale of 256 million neurosynaptic cores containing 65 billion neurons and 16 trillion synapses running only 388× slower than real time with an average spiking rate of 8.1 Hz. By using emerging PGAS communication primitives, we also demonstrate 2× better real-time performance over MPI primitives on a 4 rack Blue Gene/P (16384 CPUs, 16 TB memory).**

I. Introduction

The brain and modern computers have radically different architectures [1] suited for complementary applications. Modern computing posits a stored program model, traditionally implemented in digital, synchronous, serial, centralized, fast, hardwired, general-purpose, brittle circuits, with explicit memory addressing imposing a dichotomy between computation and data. In stark contrast, the brain uses replicated computational units of neurons and synapses implemented in mixed-mode analog-digital, asynchronous, parallel, distributed, slow, reconfigurable, specialized, and fault-tolerant biological substrates, with implicit memory addressing blurring the boundary between computation and data [2]. It is therefore no surprise that one cannot emulate the function, power, volume, and real-time performance of the brain within the modern computer architecture. This task requires a radically novel architecture.

Today, one must still build novel architectures in CMOS technology, which has evolved over the past half-century to serve modern computers and which is not optimized for delivering brain-like functionality in a compact, ultra-low-power package. For example, biophysical richness of neurons and 3D physical wiring are out of the question at the very outset. We need to shift attention from neuroscientific richness that is *sufficient* to mathematical primitives that are *necessary*. A question of profound relevance to science, technology, business, government, and society is how closely can one approximate the function, power, volume, and real-time performance of the brain within the limits of modern technology.

To this end, under the auspices of DARPA SyNAPSE, we are developing a novel, ultra-low power, compact, modular architecture called *TrueNorth* that consists of an interconnected and communicating network of extremely large numbers of neurosynaptic cores [3], [4], [5], [6]. Each core integrates computation (neurons), memory (synapses), and intra-core communication (axons), breaking the von Neumann bottleneck [7]. Each core is event-driven (as opposed to clock-driven), reconfigurable, and consumes ultra-low power. Cores operate in parallel and send unidirectional messages to one another; that is, neurons on a source core send spikes to axons on a target core. One can think of cores as gray matter canonical cortical microcircuits [8], and inter-core connectivity as long-distance white matter [9]. Like the cerebral cortex, TrueNorth is highly scalable in terms of number of cores. TrueNorth is therefore a novel non-von Neumann architecture that captures the essence of neuroscience within the limits of current CMOS technology.

As our main contribution, we describe an architectural simulator called *Compass* for TrueNorth, and demonstrate its near-perfect weak and strong scaling performance when taking advantage of native multi-threading on IBM® Blue Gene®/Q compute nodes. Compass has one-to-one equivalence to the functionality of TrueNorth.

Compass is multi-threaded, massively parallel and highly scalable, and incorporates several innovations in communication, computation, and memory. On a 16-rack IBM Blue Gene/Q supercomputer with 262144 processors and 256 TB of main memory, Compass simulated an unprecedented 256M (10^6) TrueNorth cores containing 65B (10^9) neurons and 16T (10^{12}) synapses. These results are 3× the number of estimated neurons [10] in the human cortex, comparable to the number of synapses in the monkey cortex, and 0.08× the number of synapses in the human cortex[1]. At an average neuron spiking rate of 8.1 Hz the simulation is only 388× slower than real time.

Additionally, we determine how many TrueNorth cores Compass can simulate under a soft real-time constraint. We compare approaches using both PGAS and MPI communication primitives to find the most efficient in terms of latency and

[1]The disparity between neuron and synapse scales arises because the ratio of neurons to synapses is 1 : 256 in TrueNorth, but is 1 : 10000 in the brain.

SC12, November 10-16, 2012, Salt Lake City, Utah, USA
978-1-4673-0806-9/12/$31.00 ©2012 IEEE

bandwidth. We note that one-sided inter-core communication from a neuron on one core to an axon on another core residing on a different compute node maps naturally to the PGAS model.

Compass enables large-scale simulations that provide brain-like function as a precursor to demonstrating the same functionality on TrueNorth hardware. Our goal is not to use Compass to model the brain, but to approximate brain-like function that integrates multiple sensory-motor modalities. We believe that function follows form. The form that we have chosen to concentrate on in this paper leverages the largest known long-distance wiring diagram in the macaque brain that spans cortex, thalamus, and basal ganglia [9]. Concretely, we use the macaque wiring diagram to instantiate an inter-core wiring pattern. The richness of the wiring diagram challenges the communication and computational capabilities of Compass in a manner consistent with supporting brain-like networks. To this end, we have built a novel parallel compiler that takes a high-level network description of the macaque wiring diagram and converts it to the parameters needed to configure a network of TrueNorth cores. Our compiler is sufficiently general to support other application-driven networks.

Compass is indispensable for (a) verifying TrueNorth correctness via regression testing, (b) studying TrueNorth dynamics, (c) benchmarking inter-core communication topologies, (d) demonstrating applications in vision, audition, real-time motor control, and sensor integration, (e) estimating power consumption, and (f) hypotheses testing, verification, and iteration regarding neural codes and function. We have used Compass to demonstrate numerous applications of the TrueNorth architecture, such as optic flow, attention mechanisms, image and audio classification, multi-modal image-audio classification, character recognition, robotic navigation, and spatio-temporal feature extraction.

Compass differs completely from our previous simulator, C2 [11], [12]. First, the fundamental data structure is a neurosynaptic core instead of a synapse; the synapse is simplified to a bit, resulting in 32× less storage required for the synapse data structure as compared to C2. Second, the local and long-distance anatomical connectivity in Compass that emulate, respectively, intra-core and inter-core constraints in TrueNorth have no counterpart in C2. Third, the neuron dynamics equations in Compass are amenable to efficient hardware implementation, whereas C2 focused on single-compartment phenomenological dynamic neuron models [13]. Fourth, Compass uses a fully multi-threaded programming model whereas C2 used a flat MPI programming model, rendering it incapable of exploiting the full potential of Blue Gene/Q. Last, Compass incorporates an *in situ* compiler for generating a complete TrueNorth model from a compact *CoreObject* description file, which reduces simulation set-up times by three orders of magnitude. As a result of these innovations, Compass demonstrates unprecedented weak scaling beyond the cat brain scale achieved by C2. The architecture of Compass stands in contrast to other computational neuroscientific simulators [14], [15], [16], [17], [18], [19], [20]. For reference, we point readers at reviews of large-scale brain simulations [21], [22].

Compass is a harbinger of an emerging use of today's modern supercomputers for midwifing the next generation of application-specific processors that are increasingly proliferating to satisfy a world that is hungering for increased performance and lower power while facing the projected end of CMOS scaling and increasing obstacles in pushing clock rates ever higher.

II. The TrueNorth architecture

TrueNorth is a scalable neurosynaptic computer architecture developed at IBM under the DARPA SyNAPSE program. Inspired by the organization and function of the brain, TrueNorth employs a similar vocabulary, describing CMOS circuit elements as neurons, axons, dendrites, and synapses.

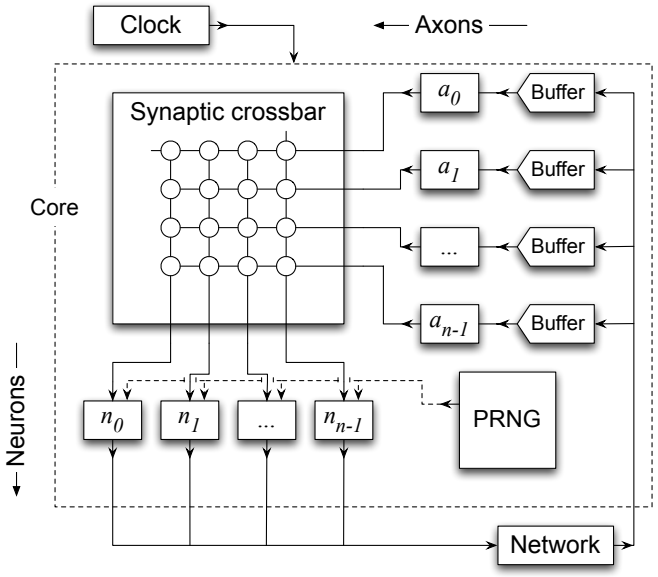


Fig. 1. Conceptual architecture of a neurosynaptic core incorporating a synaptic crossbar that contains axons (horizontal lines), dendrites (vertical lines), synapses (intersection of horizontal axons and vertical dendrites), and neurons attached to dendrites. A buffer for incoming spikes precedes each axon to account for axonal delays.

The key building block of TrueNorth is a neurosynaptic core [3], [4] as shown in figure 1, connected through a communication network to other TrueNorth cores in the system. The specific instance of a neurosynaptic core that we simulate has 256 axons, 256 dendrites feeding to 256 neurons, and a 256×256 binary crossbar synaptic array. Neurons are digital integrate-leak-and-fire circuits, characterized by configurable parameters sufficient to produce a rich repertoire of dynamic and functional behavior. A neuron on any TrueNorth core can connect to an axon on any TrueNorth core in the network; when the neuron fires it sends a spike message to the axon.

Each TrueNorth core operates in a parallel, distributed, and semi-synchronous fashion. Each TrueNorth core receives a slow clock tick at 1000 Hz to discretize neuron dynamics in 1-millisecond time steps. Otherwise, neuronal spike events drive all operations in the system. When a TrueNorth core receives

a tick from the slow clock, it cycles through each of its axons. For each axon, if the axon has a spike ready for delivery at the current time step in its buffer, each synaptic value on the horizontal synaptic line connected to the axon is delivered to its corresponding post-synaptic neuron in the TrueNorth core. If the synapse value for a particular axon-neuron pair is non-zero, then the neuron increments its membrane potential by a (possibly stochastic) weight corresponding to the axon type. After all axons are processed, each neuron applies a configurable, possibly stochastic leak, and a neuron whose membrane potential exceeds its threshold fires a spike. Each spike is then delivered via the communication network to its corresponding target axon. An axon that receives a spike schedules the spike for delivery at a future time step in its buffer. The entire cycle repeats when the TrueNorth core receives the next tick from the slow clock.

Neurons represent computation, synapses represent memory, and neuron-axon connections represent communication. Each TrueNorth core brings computation and memory into extreme proximity, breaking the von Neumann bottleneck. Note that synaptic or neuronal state never leaves a TrueNorth core and only spikes ever leave or enter any TrueNorth core. The communication network is driven solely by spike events, and requires no clocks. The neuron parameters, synaptic crossbar, and target axon for each neuron are reconfigurable throughout the system.

To ensure one-to-one equivalence between TrueNorth and Compass, we have taken judicious care in the design of both systems. For example, we have bypassed the complexity of analog neurons traditionally associated with neuromorphic systems. Also, we have adopted pseudo-random number generators with configurable seeds. As a result, Compass has become the key contract between our hardware architects and software algorithm/application designers.

III. The Compass neurosynaptic chip simulator

Compass is an architectural simulator for large-scale network models of TrueNorth cores, and is implemented using a combination of MPI library calls and OpenMP threading primitives. Compass partitions the TrueNorth cores in a model across several processes, and distributes TrueNorth cores residing in the same shared memory space within a process among multiple threads. Threads within a process independently simulate the synaptic crossbar and neuron behavior of one or more TrueNorth cores in the model. A master thread then sends any spikes destined for TrueNorth cores on remote processes in an aggregated fashion; upon reception, all threads deliver spikes to TrueNorth cores on the local process.

Compass executes its main simulation loop in a semi-synchronous manner. Listing 1 shows pseudo-code for the main simulation phases that each process in Compass executes. Each pass through the phases simulates a single tick of the global clock in a system of TrueNorth cores. In the *Synapse* phase, each process propagates input spikes from axons to neurons through the crossbar. Next, in the *Neuron* phase, each process executes the integrate, leak, and fire model for each neuron. Last, in the *Network* phase, processes communicate to send spikes from firing neurons and deliver spikes to destination axons.

```
#pragma omp shared(localBuf, remoteBuf)
#pragma omp shared(TrueNorthCores,remoteBufAgg)
#pragma omp parallel
{
    for core in TrueNorthCores[threadId] {
        // Synapse phase
        for axon in core.axons {
            axon.propagateSpike();
        }
        // Neuron phase
        for neuron in core.neurons {
            spike S := neuron.integrateLeakFire();
            if (S.dest == local)
                localBuf[threadID].push(S);
            else
                remoteBuf[threadID][S.dest].push(S);
        }
    }
#pragma omp barrier
    threadAggregate(remoteBuf, remoteBufAgg)
    if (threadID == 0) {
        for dest in remoteDests {
            if (remoteBufAgg[dest].size() != 0) {
                MPI_Isend(remoteBufAgg[dest]);
                sendCounts[dest]++;
            }
        }
    }

    // Network phase
    if (threadID == 0) {
        MPI_Reduce_Scatter(sendCounts, recvCount);
    }
    else {
        threadBuf := partition(localBuf, thread);
        deliver(spikes in threadBuf);
    }
#pragma omp barrier

#pragma omp for
    for message in recvCount messages {
#pragma critical
        {
            MPI_Iprobe(status);
            MPI_Get_count(status, recvLength);
            MPI_Recv(recvBuf, recvLength);
        } // end critical
        deliver(spikes in recvBuf);
    } // end omp for
} // end omp parallel
```

Listing 1. Pseudo-code of the main simulation loop in Compass.

To minimize communication overhead, Compass aggregates spikes between pairs of processes into a single MPI message. At startup, each process in Compass gets all destination TrueNorth core / axon pairs from the neurons within its hosted TrueNorth cores, uses an implicit TrueNorth core to process map to identify all destinations on remote processes, and preallocates per-process send buffers. During a tick, the process aggregates all spikes for TrueNorth cores at remote processes into the corresponding buffer, and uses a single MPI send call to transmit each buffer once all neurons have integrated, leaked, and fired.

In detail, each Compass process forks multiple OpenMP threads, which execute the main Compass phases in parallel as follows:

- *Synapse* phase: For each axon within a set of TrueNorth cores dedicated to a specific thread, the thread checks if a spike is ready for delivery. If a spike is ready, Compass propagates the spike along the row in the synaptic crossbar connected to the axon. For each set connection in the row, Compass buffers the spike for integration at the neuron connected to the set connection. For example, if the i^{th} axon in a TrueNorth core has a spike ready for delivery, and the ij^{th} connection is set in the crossbar, then Compass propagates the spike to the j^{th} neuron.
- *Neuron* phase: For each neuron within a TrueNorth core, threads integrate all propagated spikes, leak potential, and generate spikes as appropriate. Threads then aggregate spikes destined for TrueNorth cores at remote processes into per-process remote destination buffers (*remoteBufAgg*) so that spikes are consecutively laid out in memory for MPI message transfers. The master thread within each process lastly uses a single MPI message to send each remote destination buffer to the corresponding remote process.
- *Network* phase: The master thread uses an MPI Reduce-Scatter operation to determine how many incoming messages to expect. In parallel, Compass divides the local destination buffer uniformly among all non-master threads, each of which then delivers spikes from its portion to the destination TrueNorth cores (in detail, to the corresponding axon buffer at a specific destination axon) immediately. Performance is improved since the processing of local spikes by non-master threads overlaps with the Reduce-Scatter operation performed by the master thread.

  After the master thread completes the Reduce-Scatter operation, and after all non-master threads deliver spikes from the local buffer, all threads take turns to receive an MPI message and deliver each spike contained in the message to its destination TrueNorth core residing in the shared memory space of the Compass process. Each thread receives MPI messages in a critical section due to thread-safety issues in the MPI library [23], but delivers the spikes within the messages outside of the critical section.

IV. Parallel Compass Compiler

Compass is capable of simulating networks of tens of millions of TrueNorth cores. To build applications for such large-scale TrueNorth networks, we envisage first implementing libraries of functional primitives that run on one or more interconnected TrueNorth cores. We can then build richer applications by instantiating and connecting regions of functional primitives. Configuring such rich applications in Compass (or, for that matter, on TrueNorth hardware) can potentially require setting trillions of parameters. To make the configuration more tractable, we require tools to create lower-level core parameter specifications from higher-level, more compact descriptions of the functional regions.

We have designed and implemented a parallel processing tool called the *Parallel Compass Compiler* (PCC) that translates a compact definition of functional regions of TrueNorth cores into the explicit neuron parameter, synaptic connection parameter, and neuron-to-axon connectivity declarations required by Compass. PCC works to minimize MPI message counts within the Compass main simulation loop by assigning TrueNorth cores in the same functional region to as few Compass processes as necessary. This minimization enables Compass to use faster shared memory communication to handle most intra-region spiking, reserving more expensive MPI communication for mostly inter-region spiking. Overall, the number of Compass processes required to simulate a particular region corresponds to the computation (neuron count) and memory (synaptic connection count) required for the region, which in turn is related to the functional complexity of the region.

We show an illustrative example of three inter- and intra-connected functional regions in Figure 2. In our example, three processes manage each of the functional regions A and C, and two processes manage functional region B. For illustration clarity we present only connections of process 0 (in region A) and of process 3 (in region B) to region C. The thickness of the edges between processes reflects the strength of the connection; thicker connections represents more neuron connections.

PCC constructs the connectivity graph between functional regions using a distributed algorithm, in which each parallel PCC process compiles TrueNorth core parameters for at most one functional region. To create neuron-to-axon connections between regions, the PCC process managing the target region uses MPI message operations to send the global core ID and axon ID of an available axon to the PCC process managing the source region. To create neuron-to-axon connections within a region (for example, as with process P_0 in Figure 2), a PCC process connects cores locally; we use OpenMP threading primitives to exploit thread-level parallelism.

This exchange of information happens in an aggregated per process pair fashion. As illustrated in Figure 2, if K neurons on P_0 connect to P_5, P_5 needs to send the TrueNorth core IDs and the K axon IDs to P_0 using `MPI_Isend`. The axon types and the synaptic crossbar on the corresponding TrueNorth cores on P_5 are setup simultaneously. On P_0 the received TrueNorth core IDs and axon IDs are used to connect the source TrueNorth cores on P_0 with the target TrueNorth cores on P_5. We require a realizability mechanism for connections to guarantee that each target process has enough TrueNorth cores to satisfy incoming connection requests. One way to accomplish this is to normalize the network to have consistent neuron and axon requirements. This is equivalent to normalizing the connection matrix to have identical pre-specified column sum and row sums [24], [25], [26] - a generalization of doubly stochastic matrices. This procedure is known as iterative proportional

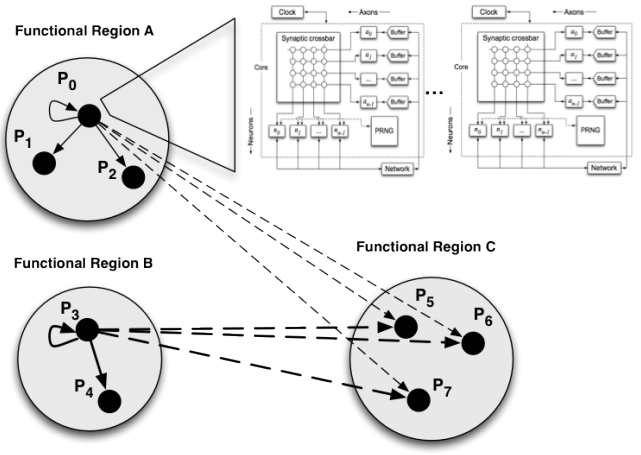


Fig. 2. Illustration of a neurosynaptic core network built using the Parallel Compass Compiler across multiple processes showing connectivity within and across functional regions.

fitting procedure (IPFP) in statistics, and as matrix balancing in linear algebra.

The high-level network description describing the network connectivity is expressed in a relatively small and compact CoreObject file. For large scale simulation of millions of TrueNorth cores, the network model specification for Compass can be on the order of several terabytes. Offline generation and copying such large files is impractical. Parallel model generation using the compiler requires only few minutes as compared to several hours to read or write it to disk. Once the compiler completes the wiring of the neurosynaptic cores across all regions, the TrueNorth cores from each processor are instantiated within Compass and the TrueNorth cores in the compiler are deallocated to free up space. Finally, Compass simulates the network model that was created.

V. CoCoMac macaque brain model network

Higher cognition in the brain is thought to emerge from the thalamocortical system, which can be divided into functionally specialized thalamic or cortical regions. Each region is composed of millions or billions of neurons, cells specialized to receive, integrate and send messages. Long range white matter connections provide communication between neurons in different regions, while short range gray matter connections provide for communication between neurons within the same region. Tremendous efforts towards mapping the brain and its connectivity have been made over decades of research, a tiny fraction of which will be drawn from here to construct a simple network that nevertheless captures several key characteristics desirable for TrueNorth simulations. In building our test network, we establish natural parallels between brain regions, white matter connectivity, gray matter connectivity, and the underlying simulator hardware. We simulate each brain region using non-overlapping sets of 1 or more processes, such that white matter communication becomes equivalent to inter-process communication, and we then establish gray matter connectivity as within process communication.

A. Structure

It is hypothesized that distinct functions are supported by signature subnetworks throughout the brain that facilitate information flow, integration, and cooperation across functionally differentiated, distributed centers. We sought to capture the richness found in these networks by specifying our test networks, modules, and the connectivity between them using a very large scale database of brain regions and connectivity in the macaque monkey called *CoCoMac* [27], [28]. Decades of white matter anatomical tracing studies in the macaque brain have been painstakingly assembled in the CoCoMac database. In previous efforts, we have integrated the contributions made to CoCoMac by many disparate labs to produce a cohesive, conflict-free network of brain regions in the macaque monkey that is three times larger than the largest previous such network [9]. Here, we reduced this network to 77 brain regions based on white matter connectivity metrics, described below, to make it more amenable to simulation. We derived volumetric information for each region from the Paxinos brain atlas [29] and accompanying software [30], which in turn was used to set relative neuron counts for each region. Volume information was not available for 5 cortical and 8 thalamic regions and so was approximated using the median size of the other cortical or thalamic regions, respectively.

B. Long range connectivity

In the CoCoMac network each brain region is represented as a vertex, and the presence of a white matter connection between two brain regions is represented as an edge between corresponding vertices. The derived network consists of 383 hierarchically organized regions spanning cortex, thalamus, and basal ganglia, and has 6,602 directed edges that capture well-known cortico-cortical, cortico-subcortical, and intra-subcortical white matter pathways [9]. In the many studies reported to CoCoMac, there are often cases where one lab reports connections for a brain region, while another lab subdivides that brain region and reports connections for one of the child subregions. For simplicity, we reduced the network by merging a child subregion into a parent region where both child and parent regions report connections. We do this by ORing the connections of the child region with that of the parent region. The smaller lower resolution network consists of 102 regions, 77 of which report connections. In the brain, the topography of connections between regions has in some cases been shown to be highly structured and focused [31] and in other cases very diffuse [32]. For a given vertex in our reduced CoCoMac graph, such focused connections correspond to a source process targeting a single process in the target region, while diffuse connections correspond to a source process targeting multiple processes in the target region (if more than one such process exists). We choose to use long range connections that are as diffuse as possible in our test network, because this places the largest burden on

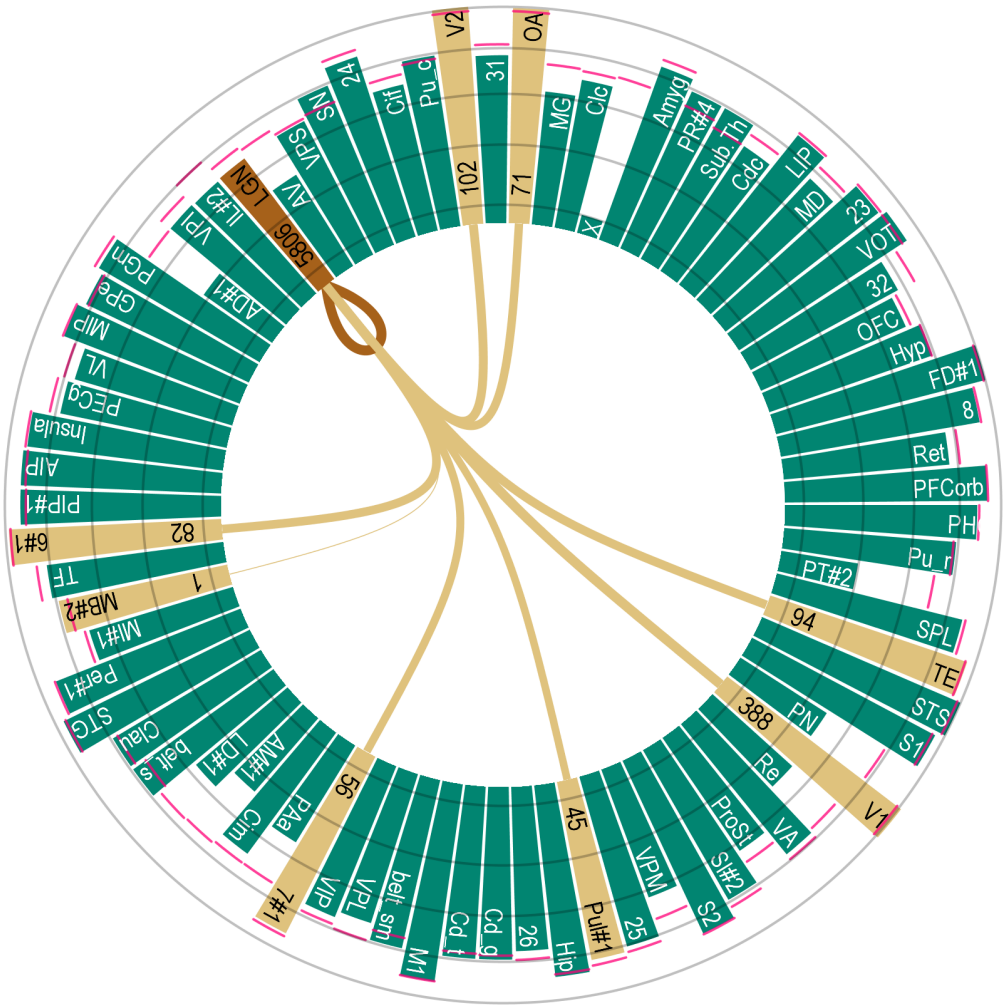


Fig. 3. Macaque brain map consisting of the 77 brain regions used for the test network. The relative number of TrueNorth cores for each area indicated by the Paxinos atlas is depicted in green, and the actual number of TrueNorth cores allocated to each region following our normalization step is depicted in red, both plotted in log space. Outgoing connections and neurons allocated in a 4096 TrueNorth cores model are shown for a typical region, LGN, which is the first stage in the thalamocortical visual processing stream.

the communication infrastructure of our simulator. As region sizes are increased through scaling, requiring more process per region, the number inter-process connections required for a given process will increase. At the same time, the number of neuron-axon connections per such link will decrease.

C. Local connectivity

Extensive studies have revealed that while each neuron can form gray matter connections to a large array of possible targets, targets are invariably within a 2-3 mm radius from the source [33]. The number of cortical neurons under a square millimeter of cortex in a the macaque brain is estimated in the tens of thousands [34], which sets an upper limit on possible neuron targets from gray matter connectivity as the nearest 3 million or so neurons, which fits within the number of neurons we are able to use per process for the simulations performed here. Therefore we choose to restrict our modeling of gray matter connectivity such that the source neuron and target neuron of gray matter connections are located on the same process. It has also been found that while each neuron often connects to the same target gray matter patches as its neighbors, this is not always the case [33]. Therefore, to provide the highest possible challenge to cache performance, we chose to ensure that all locally connecting neurons on the same TrueNorth core distribute their connections as broadly as possible across the set of possible target TrueNorth cores.

We establish our ratio of long range to local connectivity in approximately a 60/40 ratio for cortical regions, and in an 80/20 ratio for non-cortical regions. From CoCoMac we obtained a binary connection matrix. This matrix is converted to a stochastic matrix with the above gray matter percentages on the diagonals and white matter connections set to be proportional to the volume percentage of the outgoing region. Then the connection matrix was balanced as described in section IV to make the row and column sums equal to the volume of that region. The end result guarantees a realizable model in that all axon and neuron requests can be fulfilled in all regions. The brain regions used, a sample of their interconnectivity, and the results of this normalization process can be seen in figure 3.

VI. Experimental evaluation

We completed an experimental evaluation of Compass that shows near-perfect weak and strong scaling behavior when simulating a CoCoMac macaque network model on an IBM Blue Gene/Q system of 16384 to 262144 compute CPUs. We attribute this behavior to several important design and operational features of Compass that make efficient use of the communication subsystem. The Compass implementation overlaps the single collective MPI Reduce-Scatter operation in a process with the delivery of spikes from neurons destined for local cores, and minimizes the MPI communicator size by spawning multiple threads within an MPI process on multi-core CPUs in place of spawning multiple MPI processes. In operation, we observe that the MPI point-to-point messaging rate scales sub-linearly with increasing size of simulated system, due to weaker links (in terms of neuron-to-axon connections) between processes of connected brain regions. Moreover, the overall message data volume per simulated tick sent by Compass for even the largest simulated system of TrueNorth cores is well below the interconnect bandwidth of the communication subsystem.

A. Hardware platforms

We ran Compass simulations of our CoCoMac macaque network model on the IBM Blue Gene/Q, which is the third generation in the Blue Gene line of massively parallel supercomputers. Our Blue Gene/Q at IBM Rochester comprised 16 racks, with each rack in turn comprising 1024 compute nodes. Each compute node is composed of 17 processor cores on a single multi-core CPU paired with 16 GB of dedicated physical memory [35], and is connected to other nodes in a five-dimensional torus through 10 bidirectional 2 GB/second links [36]. HPC applications run on 16 of the processor cores, and can spawn up to 4 hardware threads on each core; per-node system software runs on the remaining core. The maximum available Blue Gene/Q size was therefore 16384 nodes, equivalent to 262144 application processor cores.

For all experiments on the Blue Gene/Q, we compiled with the IBM XL C/C++ compiler version 12.0 (MPICH2 MPI library version 1.4.1).

B. Weak scaling behavior

Compass exhibits nearly perfect weak scaling behavior when simulating the CoCoMac macaque network model on the IBM Blue Gene/Q. Figure 4 shows the results of experiments in which we increased the CoCoMac model size when increasing the available Blue Gene/Q CPU count, while at the same time fixing the count of simulated TrueNorth cores per node at 16384. We ran with 1 MPI process per node and 32 OpenMP threads per MPI process[2] to minimize the MPI communication overhead and maximize the available memory per MPI process. Because each Compass process distributes simulated cores uniformly across the available threads (section III), the number of simulated cores per OpenMP thread was 512 cores.

With a fixed count of 16384 TrueNorth cores per compute node, Compass runs with near-constant total wall-clock time over an increasing number of compute nodes. Figure 4(a) shows the total wall-clock time taken to simulate the CoCoMac model for 500 simulated ticks ("Total Compass"), and the wall-clock times taken per Compass execution phase in the main simulation loop (*Synapse*, *Neuron*, and *Network* in listing 1), as functions of available Blue Gene/Q CPU count. The total wall-clock time remains near-constant for CoCoMac models ranging from 16M TrueNorth cores on 16384 CPUs (1024 compute nodes) through to 256M cores on 262144 CPUs (16384 nodes); note that 256M cores is equal to 65B neurons, which is of human scale in the number of neurons. Simulating 256M cores on 262144 CPUs for 500 simulated ticks required 194 seconds[3], or 388× slower than real time.

We observe that the increase in total run time with increasing Blue Gene/Q CPU count is related primarily to an increase in communication costs during the *Network* phase in the main simulation loop. We attribute most of the increase to the time taken by the MPI Reduce-Scatter operation, which increases with increasing MPI communicator size. We also attribute some of the increase to computation and communication imbalances in the functional regions of the CoCoMac model.

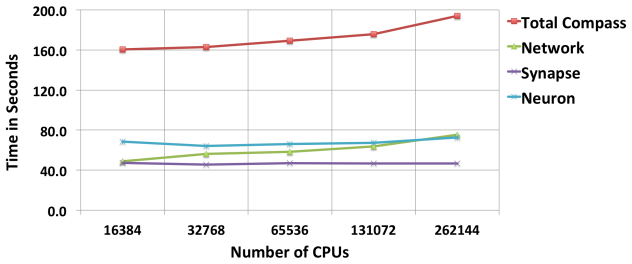
We see that some of the increase in total run time arises from an increase in the MPI message traffic during the *Neuron* phase. Figure 4(b) shows the MPI message count and total spike count (equivalent to the sum of white matter spikes from all MPI processes) per simulated tick as functions of the available Blue Gene/Q CPU count. As we increase the CoCoMac model size in the number of TrueNorth cores, more MPI processes represent the 77 macaque brain regions to which spikes can be sent, which leads to an increase in the message count and volume. We note, however, that the increase in message count is sub-linear, given that the white matter connections become thinner and therefore less frequented with increasing model size, as discussed in section V-B. For a system of 256M cores on 262144 CPUs, Compass generates and sends approximately 22M spikes per simulated tick, which at 20 bytes per spike equals 0.44 GB per tick, and is well below the 5D torus link bandwidth of 2 GB/s.

C. Strong scaling behavior

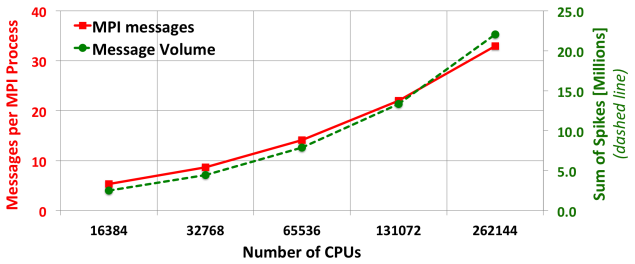
As with weak scaling, Compass exhibits excellent strong scaling behavior when simulating the CoCoMac macaque network model on the IBM Blue Gene/Q. Figure 5 shows the results of experiments in which we fixed the CoCoMac model size at 32M TrueNorth cores (8.2B neurons) while increasing the available Blue Gene/Q CPU count. As with the weak scaling results in figure 4(a), we show the total wall-clock time

[2]We are investigating unexpected system errors that occurred when running with the theoretical maximum of 64 OpenMP threads per MPI process.

[3]We exclude the CoCoMac compilation times from the total time. For reference, compilation of the 256M core model using the PCC (section IV) required 107 wall-clock seconds, mostly due to the communication costs in the white matter wiring phase.



(a) Compass total runtime and breakdown into major components



(b) Compass messaging and data transfer analysis, per simulation step

Fig. 4. Compass weak scaling performance when simulating the CoCoMac macaque network model on an IBM Blue Gene/Q, with a fixed count of 16384 TrueNorth cores per Blue Gene/Q compute node. Each Blue Gene/Q rack had 16384 compute CPUs (1024 compute nodes). We ran with one MPI process per node and 32 OpenMP threads per MPI process.

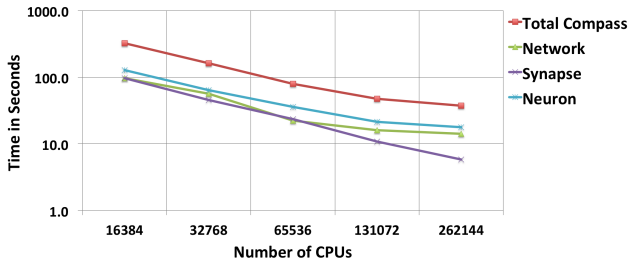


Fig. 5. Compass strong scaling performance when simulating a fixed CoCoMac macaque network model of 32M TrueNorth cores on an IBM Blue Gene/Q. Each Blue Gene/Q rack had 16384 compute CPUs (1024 compute nodes). We ran with one MPI process per node and 32 OpenMP threads per MPI process.

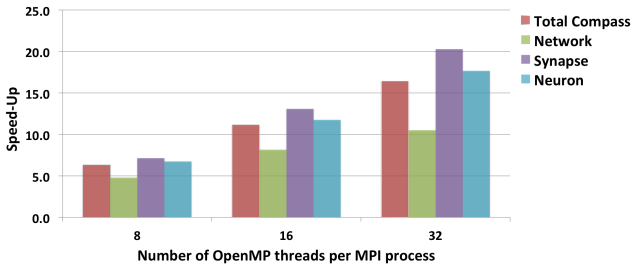


Fig. 6. OpenMP multi-threaded scaling tests when simulating a fixed CoCoMac macaque network model of 64M TrueNorth cores on a 65536 CPU (four rack) IBM Blue Gene/Q with one MPI process per compute node. We show the speed-up obtained with increasing numbers of threads over a baseline of one MPI process per node with one OpenMP thread (and therefore with 15 of 16 CPU cores idled per node).

taken to simulate the CoCoMac model for 500 simulated ticks and the wall-clock times taken per Compass execution phase in the main simulation loop. Simulating 32M cores takes 324 seconds on 16384 Blue Gene/Q CPUs (1 rack; the baseline), 47 seconds on 131073 CPUs (8 racks; a speed-up of 6.9× over the baseline with 8× more computation and memory capacity), and 37 seconds on 262144 CPUs (16 racks; a speed-up of 8.8× over the baseline with 16× more computation and memory capacity). We observe that perfect scaling is inhibited by the communication-intense main loop phases when scaling from 131072 CPUs (8 racks) up to 262144 CPUs (16 racks).

D. Thread scaling behavior

Compass obtains excellent multi-threaded scaling behavior when executing on the IBM Blue Gene/Q. Figure 6 shows the results of experiments in which we increase the number of OpenMP threads per MPI process, while at the same time fixing the CoCoMac macaque network model size at 64M TrueNorth cores. We show the speed-up over the baseline in the total wall-clock time taken to simulate the CoCoMac model for 500 simulated ticks and the speed-up in the wall-clock times taken per Compass execution phase in the main simulation loop, where the baseline is the time taken when each MPI process spawns only one OpenMP thread. We do not quite achieve perfect scaling in the number of OpenMP threads due to a critical section in the *Network* phase that creates a serial bottleneck at all thread counts.

In other multi-threaded experiments, we found that trading off between the number of MPI processes per compute node and the number of OpenMP threads per MPI processes yielded little change in performance. For example, simulation runs of Compass with one MPI process per compute node and 32 OpenMP threads per process achieved nearly similar performance to runs with 16 MPI processes per compute node and 2 OpenMP threads per process. Using fewer MPI processes and more OpenMP processes per thread reduces the size of the MPI communicator for the MPI Reduce-Scatter operation in the *Network* phase, which reduces the time taken by the phase. We observe, however, that this performance improvement is offset by false sharing penalties in the CPU caches due to increased size of the shared memory region for the threads in each MPI process.

VII. Comparison of the PGAS and MPI Communication Models for Real-Time Simulation

In addition to evaluating an implementation of Compass using the MPI message passing library (section III), we evaluated a second implementation based on the Partitioned Global Address Space (PGAS) model to explore the benefits of one-sided communications. We achieved clear real-time simulation performance improvements with the PGAS-based implementation over the MPI-based implementation. For this comparison,

we ran both PGAS and MPI versions of Compass on a system of four 1024-node IBM Blue Gene/P racks installed at IBM Watson, as no C/C++ PGAS compiler yet exists for the Blue Gene/Q. Each compute node in the Blue Gene/P comprises 4 CPUs with 4 GB of dedicated physical memory. We compiled with the IBM XL C/C++ compiler version 9.0 (MPICH2 MPI library version 1.0.7). For PGAS support, we reimplemented the Compass messaging subroutines using Unified Parallel C (UPC), and compiled with the Berkeley UPC compiler version 2.14.0 [37], [38]; the Berkeley UPC runtime uses the GASNet library [39] for one-sided communication. Both GASNet and the MPI library implementation on Blue Gene/P build upon the Deep Computing Messaging Framework (DCMF) [40] to achieve the best available communication performance.

A. Tuning for real-time simulation

Real-time simulation—1 millisecond of wall-clock time per 1 millisecond of simulated time—is important for designing applications on the TrueNorth architecture. Real-time simulations enable us to implement and test applications targeted for TrueNorth cores in advance of obtaining the actual hardware, and to debug and cross-check the TrueNorth hardware designs. Such simulations, however, already require that we reduce the size of the simulated system of TrueNorth cores in order to reduce the total per-tick latency of the *Synapse* and *Neuron* phases. We looked for further performance improvements in the *Network* phase to maximize the size of the simulated system, given that the *Network* phase incurs the bulk of the latency in the main simulation loop.

The PGAS model of one-sided messaging is better suited to the behavior of simulated TrueNorth cores in Compass than the MPI model is. The source and ordering of spikes arriving at an axon during the *Network* phase in a simulated tick do not affect the computations during the *Synapse* and *Neuron* phases of the next tick. Therefore, each Compass process can use one-sided message primitives to insert spikes in a globally-addressable buffer residing at remote processes, without incurring either the overhead of buffering those spikes for sending, or the overhead of tag matching when using two-sided MPI messaging primitives. Further, using one-sided message primitives enables the use of a single global barrier with very low latency to synchronize writing into the globally-addressable buffers with reading from the buffers, instead of needing a collective Reduce-Scatter operation that scales linearly with communicator size. We did not attempt to use MPI-2 one-sided messaging primitives, as Hoefler et al. have demonstrated that such an approach is not promising from a performance standpoint [41].

We experimented with writing our own custom synchronization primitives to provide a local barrier for subgroups of processes. In this way, each process only synchronizes with other processes with which it sends or receives spikes, as opposed to each process synchronizing with all other processes. In practice, though, we found that the Berkeley UPC synchronization primitives, which are built upon the fast DCMF native barriers, outperformed our custom primitives.

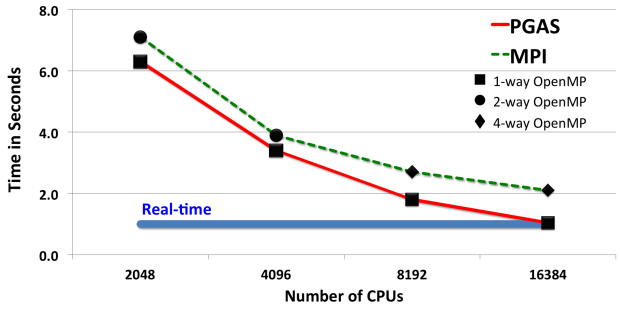


Fig. 7. Results comparing the PGAS and MPI communication models for real-time simulations in Compass over a Blue Gene/P system of four racks (16384 CPUs). We simulated 81K TrueNorth cores for 1000 ticks, with neurons firing on average at 10 Hz. For each size of Blue Gene/P, we report the result for each implementation using the best-performing thread configuration.

B. Real-time simulation results

To compare the real-time simulation performance of the PGAS implementation of Compass to the MPI implementation, we ran a set of strong scaling experiments using a synthetic system of TrueNorth cores simulated over four 1024-node Blue Gene/P racks comprising 16384 CPUs. We began by finding the largest size of system we could simulate in real time on all four racks, and then measured the time taken to simulate the same system for 1000 ticks on progressively fewer racks. For the synthetic system[4], 75% of the neurons in each TrueNorth core connect to TrueNorth cores on the same Blue Gene/P node, while the remaining 25% connect to TrueNorth cores on other nodes. All neurons fire on average at 10 Hz.

Our results show that the PGAS implementation of Compass is able to simulate the synthetic system with faster run times than the MPI implementation. Figure 7 shows the strong scaling experimental results. The PGAS implementation is able to simulate 81K TrueNorth cores in real time (1000 ticks in one second) on four racks, while the MPI implementation takes 2.1× as long (1000 ticks in 2.1 seconds). The benefits of PGAS arise from the latency advantages of PGAS over MPI on the Blue Gene/P [38] and the elimination of the MPI Reduce-Scatter operation.

For each experiment, we only report the result for each implementation using the best-performing thread configuration. Thus, over four racks (16384 CPUs) of Blue Gene/P, we show the result for the MPI implementation with one MPI process (each having four threads) per node, while on one rack (4096 CPUs) of Blue Gene/P we show the result for the MPI implementation with two MPI processes (each having two threads) per node. For all configuration, we show the result for the PGAS implementation with four UPC instances (each having one thread) per node.

[4]We do not use the CoCoMac model for real-time simulations because the size of simulated system has insufficient TrueNorth cores to populate each CoCoMac region.

VIII. Conclusions

Cognitive Computing [2] is the quest for approximating the mind-like function, low power, small volume, and real-time performance of the human brain. To this end, we have pursued neuroscience [42], [43], [9], nanotechnology [3], [4], [44], [5], [6], and supercomputing [11], [12]. Building on these insights, we are marching ahead to develop and demonstrate a novel, ultra-low power, compact, modular, non-von Neumann, cognitive computing architecture, namely, TrueNorth. To set sail for TrueNorth, in this paper, we reported a multi-threaded, massively parallel simulator, Compass, that is functionally equivalent to TrueNorth. As a result of a number of innovations in communication, computation, and memory, Compass demonstrates unprecedented scale with its number of neurons comparable to the human cortex and number of synapses comparable to the monkey cortex while achieving unprecedented time-to-solution. Compass is a Swiss-army knife for our ambitious project supporting all aspects from architecture, algorithms, to applications.

TrueNorth and Compass represent a massively parallel and distributed architecture for computation that complement the modern von Neumann architecture. To fully exploit this architecture, we need to go beyond the "intellectual bottleneck" [7] of long, sequential programming to short, parallel programming. To this end, our next step is to develop a new parallel programming language with compositional semantics that can provide application and algorithm designers with the tools to effectively and efficiently unleash the full power of the emerging new era of computing. Finally, TrueNorth is designed to best approximate the brain within the constraints of modern CMOS technology, but, now, informed by TrueNorth and Compass, we ask: how might we explore an entirely new technology that takes us beyond Moore's law, and beyond the ever-increasing memory-processor latency, beyond need for ever-increasing clock rates to a new era of computing? Exciting!

Acknowledgments

This research was sponsored by DARPA under contract No. HR0011-09-C-0002. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressly or implied, of DARPA or the U.S. Government.

We thank Filipp Akopyan, John Arthur, Andrew Cassidy, Bryan Jackson, Rajit Manohar, Paul Merolla, Rodrigo Alvarez-Icaza, and Jun Sawada for their collaboration on the TrueNorth architecture, and our university partners Stefano Fusi, Rajit Manohar, Ashutosh Saxena, and Giulio Tononi as well as their research teams for their feedback on the Compass simulator. We are indebted to Fred Mintzer for access to IBM Blue Gene/P and Blue Gene/Q at the IBM T.J. Watson Research Center and to George Fax, Kerry Kaliszewski, Andrew Schram, Faith W. Sell, Steven M. Westerbeck for access to IBM Rochester Blue Gene/Q, without which this paper would have been impossible. Finally, we would like to thank David Peyton for his expert assistance revising this manuscript.

References

[1] J. von Neumann, *The Computer and The Brain*. Yale University Press, 1958.

[2] D. S. Modha, R. Ananthanarayanan, S. K. Esser, A. Ndirango, A. J. Sherbondy, and R. Singh, "Cognitive computing," *Communications of the ACM*, vol. 54, no. 8, pp. 62–71, 2011.

[3] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, and D. Modha, "A digital neurosynaptic core using embedded crossbar memory with 45pj per spike in 45nm," in *Custom Integrated Circuits Conference (CICC), 2011 IEEE*, sept. 2011, pp. 1 –4.

[4] J. Seo, B. Brezzo, Y. Liu, B. Parker, S. Esser, R. Montoye, B. Rajendran, J. Tierno, L. Chang, D. Modha, and D. Friedman, "A 45nm cmos neuromorphic chip with a scalable architecture for learning in networks of spiking neurons," in *Custom Integrated Circuits Conference (CICC), 2011 IEEE*, sept. 2011, pp. 1 –4.

[5] N. Imam, F. Akopyan, J. Arthur, P. Merolla, R. Manohar, and D. S. Modha, "A digital neurosynaptic core using event-driven qdi circuits," in *ASYNC 2012: IEEE International Symposium on Asynchronous Circuits and Systems*, 2012.

[6] J. V. Arthur, P. A. Merolla, F. Akopyan, R. Alvarez-Icaza, A. Cassidy, S. Chandra, S. K. Esser, N. Imam, W. Risk, D. Rubin, R. Manohar, and D. S. Modha, "Building block of a programmable neuromorphic substrate: A digital neurosynaptic core," in *International Joint Conference on Neural Networks*, 2012.

[7] J. W. Backus, "Can programming be liberated from the von neumann style? a functional style and its algebra of programs." *Communications of the ACM*, vol. 21, no. 8, pp. 613–641, 1878.

[8] V. B. Mountcastle, *Perceptual Neuroscience: The Cerebral Cortex*. Harvard University Press, 1998.

[9] D. S. Modha and R. Singh, "Network architecture of the long distance pathways in the macaque brain," *Proceedings of the National Academy of the Sciences USA*, vol. 107, no. 30, pp. 13 485–13 490, 2010. [Online]. Available: http://www.pnas.org/content/107/30/13485.abstract

[10] C. Koch, *Biophysics of Computation: Information Processing in Single Neurons*. Oxford University Press, New York, New York, 1999.

[11] R. Ananthanarayanan and D. S. Modha, "Anatomy of a cortical simulator," in *Supercomputing 07*, 2007.

[12] R. Ananthanarayanan, S. K. Esser, H. D. Simon, and D. S. Modha, "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*, ser. SC '09. New York, NY, USA: ACM, 2009, pp. 63:1–63:12. [Online]. Available: http://doi.acm.org/10.1145/1654059.1654124

[13] E. M. Izhikevich, "Which model to use for cortical spiking neurons," *IEEE Transactions on Neural Networks*, vol. 15, pp. 1063–1070, 2004. [Online]. Available: http://www.nsi.edu/users/izhikevich/publications/whichmod.pdf

[14] H. Markram, "The Blue Brain Project," *Nature Reviews Neuroscience*, vol. 7, no. 2, pp. 153–160, Feb. 2006. [Online]. Available: http://dx.doi.org/10.1038/nrn1848

[15] H. Markram, "The Human Brain Project," *Scientific American*, vol. 306, no. 6, pp. 50–55, Jun. 2012.

[16] R. Brette and D. F. M. Goodman, "Vectorized algorithms for spiking neural network simulation," *Neural Comput.*, vol. 23, no. 6, pp. 1503–1535, 2011. [Online]. Available: http://dx.doi.org/10.1162/NECO_a_00123

[17] M. Djurfeldt, M. Lundqvist, C. Johansson, M. Rehn, O. Ekeberg, and A. Lansner, "Brain-scale simulation of the neocortex on the ibm blue gene/l supercomputer," *IBM J. Res. Dev.*, vol. 52, no. 1/2, pp. 31–41, Jan. 2008. [Online]. Available: http://dl.acm.org/citation.cfm?id=1375990.1375994

[18] E. M. Izhikevich and G. M. Edelman, "Large-scale model of mammalian thalamocortical systems," *Proceedings of the National Academy of Sciences*, vol. 105, no. 9, pp. 3593–3598, Mar. 2008. [Online]. Available: http://dx.doi.org/10.1073/pnas.0712231105

[19] J. M. Nageswaran, N. Dutt, J. L. Krichmar, A. Nicolau, and A. Veidenbaum, "Efficient simulation of large-scale spiking neural networks using cuda graphics processors," in *Proceedings of the 2009 international joint conference on Neural Networks*, ser. IJCNN'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 3201–3208. [Online]. Available: http://dl.acm.org/citation.cfm?id=1704555.1704736

[20] M. M. Waldrop, "Computer modelling: Brain in a box," *Nature*, vol. 482, pp. 456–458, 2012. [Online]. Available: http://dx.doi.org/10.1038/482456a

[21] H. de Garis, C. Shuo, B. Goertzel, and L. Ruiting, "A world survey of artificial brain projects, part i: Large-scale brain simulations," *Neurocomput.*, vol. 74, no. 1-3, pp. 3–29, Dec. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.neucom.2010.08.004

[22] R. Brette, M. Rudolph, T. Carnevale, M. Hines, D. Beeman, J. M. Bower, M. Diesmann, A. Morrison, P. H. Goodman, A. P. Davison, S. E. Boustani, and A. Destexhe, "Simulation of networks of spiking neurons: A review of tools and strategies," *Journal of Computational Neuroscience*, vol. 2007, pp. 349–398, 2007.

[23] D. Gregor, T. Hoefler, B. Barrett, and A. Lumsdaine, "Fixing Probe for Multi-Threaded MPI Applications," Indiana University, Tech. Rep. 674, Jan. 2009.

[24] R. Sinkhorn and P. Knopp, "Concerning nonnegative matrices and doubly stochastic matrices." *Pacific Journal of Mathematics*, vol. 21, no. 2, pp. 343–348, 1967.

[25] A. Marshall and I. Olkin, "Scaling of matrices to achieve specified row and column sums," *Numerische Mathematik*, vol. 12, pp. 83–90, 1968, 10.1007/BF02170999. [Online]. Available: http://dx.doi.org/10.1007/BF02170999

[26] P. Knight, "The sinkhorn-knopp algorithm: convergence and applications," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 1, pp. 261–275, 2008.

[27] R. Kötter, "Online retrieval, processing, and visualization of primate connectivity data from the CoCoMac database," *Neuroinformatics*, vol. 2, pp. 127–144, 2004.

[28] "Cocomac (collations of connectivity data on the macaque brain)," www.cocomac.org.

[29] G. Paxinos, X. Huang, and M. Petrides, *The Rhesus Monkey Brain in Stereotaxic Coordinates*. Academic Press, 2008. [Online]. Available: http://books.google.co.in/books?id=7HW6HgAACAAJ

[30] B. G. Kötter R., Reid A.T., "An introduction to the cocomac-paxinos-3d viewer," in *The Rhesus Monkey Brain in Stereotaxic Coordinates*, G. Paxinos, X.-F. Huang, M. Petrides, and A. Toga, Eds. Elsevier, San Diego., 2008, ch. 4.

[31] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *J. Physiol.*, vol. 160, no. 1, pp. 106–154, 1962.

[32] A. Peters and E. G. Jones, *Cerebral Cortex. Vol 4. Association and auditory cortices*. Springer, 1985.

[33] V. B. Mountcastle, "The columnar organization of the neocortex," *Brain*, vol. 120, no. 4, pp. 701–22, 1997.

[34] C. E. Collins, D. C. Airey, N. A. Young, D. B. Leitch, and J. H. Kaas, "Neuron densities vary across and within cortical areas in primates," *Proceedings of the National Academy of Sciences*, vol. 107, no. 36, pp. 15 927–15 932, 2010.

[35] R. A. Haring, M. Ohmacht, T. W. Fox, M. K. Gschwind, D. L. Satterfield, K. Sugavanam, P. W. Coteus, P. Heidelberger, M. A. Blumrich, R. W. Wisniewski, A. Gara, G. L.-T. Chiu, P. A. Boyle, N. H. Chist, and C. Kim, "The IBM Blue Gene/Q Compute Chip," *IEEE Micro*, vol. 32, pp. 48–60, 2012.

[36] D. Chen, N. A. Eisley, P. Heidelberger, R. M. Senger, Y. Sugawara, S. Kumar, V. Salapura, D. L. Satterfield, B. Steinmacher-Burow, and J. J. Parker, "The IBM Blue Gene/Q interconnection network and message unit," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '11. New York, NY, USA: ACM, 2011, pp. 26:1–26:10. [Online]. Available: http://doi.acm.org/10.1145/2063384.2063419

[37] "The Berkeley UPC Compiler," 2002, http://upc.lbl.gov.

[38] R. Nishtala, P. H. Hargrove, D. O. Bonachea, and K. A. Yelick, "Scaling communication-intensive applications on BlueGene/P using one-sided communication and overlap," in *Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing*, ser. IPDPS '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 1–12. [Online]. Available: http://dx.doi.org/10.1109/IPDPS.2009.5161076

[39] D. Bonachea, "GASNet Specification, v1.1," University of California at Berkeley, Berkeley, CA, USA, Tech. Rep., 2002.

[40] S. Kumar, G. Dozsa, G. Almasi, P. Heidelberger, D. Chen, M. E. Giampapa, M. Blocksome, A. Faraj, J. Parker, J. Ratterman, B. Smith, and C. J. Archer, "The Deep Computing Messaging Framework: Generalized Scalable Message Passing on the Blue Gene/P Supercomputer," in *Proceedings of the 22nd annual international conference on Supercomputing*, ser. ICS '08. New York, NY, USA: ACM, 2008, pp. 94–103. [Online]. Available: http://doi.acm.org/10.1145/1375527.1375544

[41] T. Hoefler, C. Siebert, and A. Lumsdaine, "Scalable Communication Protocols for Dynamic Sparse Data Exchange," in *Proceedings of the 2010 ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP'10)*. ACM, Jan. 2010, pp. 159–168.

[42] D. S. Modha, "A conceptual cortical surface atlas," *PLoS ONE*, vol. 4, no. 6, p. e5693, 2009.

[43] A. J. Sherbondy, R. Ananthanrayanan, R. F. Dougherty, D. S. Modha, and B. A. Wandell, "Think global, act local; projectome estimation with bluematter," in *Proceedings of MICCAI 2009*. Lecture Notes in Computer Science, 2009.

[44] B. L. Jackson, B. Rajendran, G. S. Corrado, M. Breitwisch, G. W. Burr, R. Cheek, K. Gopalakrishnan, S. Raoux, C. T. Rettner, A. G. Schrott, R. S. Shenoy, B. N. Kurdi, C. H. Lam, and D. S. Modha, "Nanoscale electronic synapses using phase change devices," *ACM Journal on Emerging Technologies in Computing Systems, forthcoming*, 2012.