Introducing Temporal Behavior to Computing Science

János Végh

1 Introduction

Computing science is on the border of mathematics and, through its physical implementation, science. Since the beginning of computing, the computing paradigm itself, "*the implicit hardware/software contract* [1]", defined how mathematics-based theory and its science-based implementation must cooperate. Mathematics, however, considers only the *dependencies* between its operands; it assumes that the needed operands are instantly available. That is, computing science considers that performing operations, delivering operands to and from processing units, is as kind of engineering imperfectness. At the time when von Neumann proposed his famous abstraction, both time of processing and time of accessing data (including those on a mass storage device) were in the milliseconds region, while physical data delivery time was in the range of microseconds, i.e., three orders of magnitude smaller. *It was a plausible assumption to consider that total time of processing comprises only time of computation plus time of data access; data delivery time was neglected.*

For today, however, technical development changed the relations between those timings drastically. Today the data access time is much larger than the time needed to process them. Besides, the relative weight of the data transfer time has grown tremendously, for many reasons. Firstly, miniaturizing the processors to sub-millimeter size, while keeping the rest of the components (such as buses) above the centimeter scale. Secondly, the single-processor performance stalled [2],

Project no. 136496 has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the K funding scheme.

J. Végh (✉)
Kalimános BT, Debrecen, Hungary

H. R. Arabnia et al. (eds.), *Advances in Software Engineering, Education, and e-Learning*, Transactions on Computational Science and Computational Intelligence, https://doi.org/10.1007/978-3-030-70873-3_33

mainly because of reaching the limits, the laws of nature enable [3]. Thirdly, making truly parallel computers failed [1], and to reach the needed high computing performance we need to put together an excessive number of segregated processors. This latter way replaces *parallel computing* with *parallelized sequential computing*, disregarding that the operating rules of that different kind of computing [4–6] sharply differ from those of the segregated processors. Fourthly, the mode of utilization (mainly multitasking) forced out using operating system (OS), which imitates a "new processor" for a new task, at serious time expenses. Finally, the idea of "real-time connected everything" introduced geographically large distances with the corresponding several millisecond data delivery times. Theory of computing kept the idea of "instant delivery"; although even within the core, wiring has an increasing role. The idea of non-temporal behavior was confirmed by accepting "weak scaling" [7], suggesting that *all housekeeping times, such as organizing joint work of parallelized serial processors, sharing resources, using exceptions and OS services, delivering data between processing units and data storage units, are negligible*.

Vast computing systems can cope with their tasks with growing difficulty, enormously decreasing computing efficiency, and enormously growing energy consumption; one can experience similar issues in the world of networked edge devices. Being not aware of that collaboration between processors needs a different approach (another paradigm), resulted in demonstrative failures already known (such as supercomputers Gyoukou and Aurora'18, or brain simulator SpiNNaker)[1] and many more may follow: such as Aurora'21 [9], the China mystic supercomputers[2] and the EU planned supercomputers.[3] General-purpose computing systems comprising "only" millions of processors already show the issues, and brain-like systems want to comprise four orders of magnitude higher number of computing elements. When targeting neuromorphic features such as "deep learning training", the issues start to manifest already at a couple of dozens of processors [10, 11]. The scaling is nonlinear [5], strongly depending on the workload type, and the Artificial Intelligence (AI)-class workload is one of the worst workloads [5, 11] one can run on conventional architectures.[4]

"*Successfully addressing these challenges [of neuromorphic computing] will lead to a new class of computers and systems architectures*" [12]. However, the roundtable concentrated *only* on finding new materials and different gate devices. *They did not even mention that for such systems new computing paradigm may*

[1]The explanations are quite different: Gyoukou was withdrawn after its first appearance; Aurora failed: retargeted and delayed; Despite the failure of SpiNNaker1, the SpiNNaker2 is also under construction [8]; "Chinese decision-makers decided to withhold the country's newest Shuguang supercomputers even though they operate more than 50 percent faster than the best current US machines".

[2]https://www.scmp.com/tech/policy/article/3015997/china-has-decided-not-fan-flames-super-computing-rivalry-amid-us.

[3]https://ec.europa.eu/newsroom/dae/document.cfm?doc_id=60156.

[4]https://www.nextplatform.com/2019/10/30/cray-revamps-clusterstor-for-the-exascale-era/: artificial intelligence, ... it's the most disruptive workload from an I/O pattern perspective.

also be needed. The result was that, as noticed by judges of the Gordon Bell Prize, *"surprisingly, [among the winners of the supercomputer competition] there have been no brain-inspired massively parallel specialized computers"* [13]. Despite the vast need and investments, furthermore the concentrated and coordinated efforts, just because of the vital bottleneck: ***the missing theory***.

2 Introducing Time to Computing

As suspected by many experts, the computing paradigm itself, "*the implicit hardware/software contract* [1]", is responsible for the experienced issues: "*No current programming model is able to cope with this development [of processors], though, as they essentially still follow the classical van Neumann model*" [14]. When thinking about "advances beyond 2020", the solution was expected from the "*more efficient implementation of the von Neumann architecture*" [15], however.

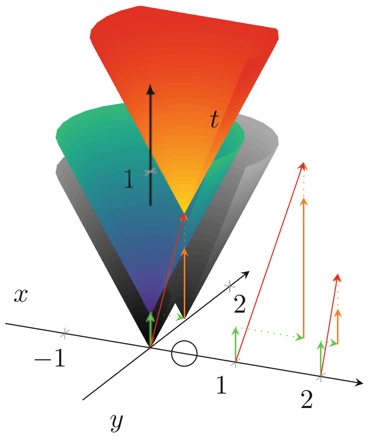
There are many analogies between science and computing [16]; among others, how they handle time. Both classic science and classic computing assume instant (infinitely quick) interaction between its objects. That is, an event happening at any location can be instantly seen at all other locations: time has no specific role, and an event has immediate effect on all other considered objects. In science, discovering that the speed of light is insurmountable, led to introducing the *four-dimensional space-time*. Special relativity introduces a 'fourth space dimension', and *we calculate that coordinate of the Minkowski space from the time as the distance the light traverses in a given time*.

2.1 Why Temporal Logic Is Needed

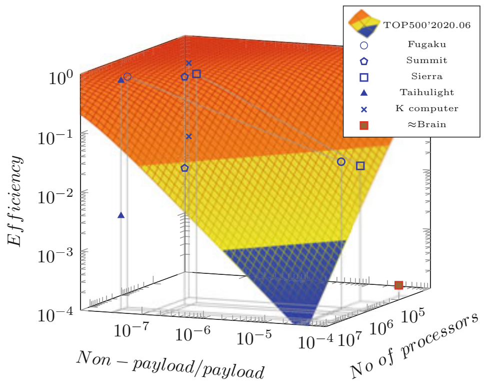
In computing, distances get defined during fabrication of components and assembling the system. In biological systems, nature defines neuronal distances, and in 'wet' neuro-biology, signal timing rather than axon length is the right (measurable) parameter. To describe temporal operation of computing systems correctly, *we need to find out how much later a component notices that an event occurred in the system*. To introduce a *temporal logic* (i.e. that value of a logical expression depends on where and when it is evaluated) into computing, the *reverse* of Minkowski transform is required: we need to use a special 4-vector, where all coordinates are time values: the first three are the corresponding local coordinates (distances from the location of the event, divided by the speed of interaction) having time dimension, and the fourth coordinate is the time itself; that is, we introduce a *4 dimensional time-space* system. The resemblance with the Minkowski-space is obvious, and the name difference signals the different aspects of utilization.

Figure 1a shows *why time must be considered explicitly in all kinds of computing*. The figure shows (for visibility) a 3-dimensional coordinate system: how an event behaves in a two-dimensional space plus time (the concept is easier to visualize with the number of spatial dimensions reduced from three to two). In the figure, the direction ‘y’ is not used, but enables to place observers at the same distance from the event, without the need to locate them in the same point. The event happens at point (0,0,0), the observers are located on the ‘x’ axis; the vertical scale corresponds to the time.

In the classic physical hypothetical experiment, we switch on a light in the origo, and the observer switches his light when notices that the first light was switched on. If we graph the growing circle with the vertical axis of the graph representing time, the result is a cone, known as the *future light cone* (in 2D space plus a time dimension). Both light sources have some “processing time”’, that passes between noticing the light (receiving the instruction) and switching the light on (performing the instruction). That is, the instruction is received at the origo, at the bottom of the green arrow. The light goes on at the head of the arrow, (i.e., at the same location, but at a later time), after that the ‘processing time’ T_p passed. Following that, the light propagates in the two spatial dimensions as a circle around axis “t”. Observers at larger distance notice the light at a later time: a ‘transmission time’ T_t is needed. If “processing time” of the light source of the first event were zero, the light would propagate along the gray surface at the origo. However, because of the finite processing time, the light will propagate along the blueish cone surface, at the head of the green arrow.



(a)



(b)

Fig. 1 The origin of “idle waiting time” and its effect on the efficiency on parallelized sequential processing systems. (**a**) The computing operation in time-space approach. The processing operators can be gates, processors, neurons or networked computers. (**b**) The surface and the figure marks show at what efficiency the top supercomputers run the ‘best workload’ benchmark HPL, and the ‘real-life load’ HPCG [6]. The right bottom part displays the expected efficiency [17] of running neuromorphic calculations on SPA computers

A circle denotes position of our observer on the axis “x”. With zero “transmission time”, the second gray conical surface (at the head of the green dotted arrow) would describe his light. However, its “processing time” can only begin when the observer notices the light at his position: when the dotted red arrow hits the blueish surface. At that point begins “processing time” of the second light source; the yellowish conical surface describes the second light propagation. The horizontal (green dotted) arrow describes the physical distance of the observer (as a time coordinate), the vertical (red dotted) arrow describes the time delay of the observer light. It comprises two components: the T_t transmission time to the observer and its T_p processing time. The light cone of the observer starts at $t = 2 * T_p + T_t$.

The red arrow represents the resulting *apparent processing time* T_A: the longer is the red vector; the slower is the system. As the vectors are in the same plane, $T_A = \sqrt{T_t^2 + (2 \cdot T_p + T_t)^2}$, that is $T_A = T_p \cdot \sqrt{R^2 + (2 + R)^2}$. This means, that *the apparent time is a non-linear function of both of its component times* and *their ratio* R. If more computing elements are involved, T_t denotes the longest transmission time. (Similar statement is valid if the T_p times are different) The effect is significant: if $R = 1$, the apparent execution time of performing the two computations is more than 3 times longer than the processing time. Two more observers are located on the axis ‘x’, at the same position. For visibility, their timings are displayed at points ‘1’ and ‘2’, respectively. Their results illustrate the influence of the transmission speed (and/or the ratio R). In their case *the transmission speed differs by a factor of two* compared to that displayed at point ‘0’; in this way three different $R = T_t/T_p$ ratios are displayed.

Notice that at half transmission speed (the horizontal green arrow is twice as long as that in the origo) the vector is considerably longer, while at double transmission speed, the decrease of the time is much less expressed.[5] Given that the *apparent processing time* T_A defines the performance of the system, T_p and T_t must be concerted.

2.2 *Consequences of Temporal Behaviour*

Notice an important aspect: *the* T_p *transmission time is an ‘idle time’* (the orange arrow on the figure) for the observer: it is ready to run, takes power, but does no useful work. Due to their finite physical size and limited interaction speed (both neglected in the classic paradigm), *temporal operation of computing systems results inherently in an idle time of their processing units*,[6] and—since it sensitively depends on many factors and conditions—*can be a significant contributor to non-payload portion of their processing time*. With other major contributors, originating

[5]Reference [6] discusses this phenomenon in details.

[6]It can be a crucial factor of inefficiency of general-purpose chips [18].

Listing 1. The essential lines of source code of the one-bit adder implemented in SystemC

```
//We are making a 1-bit addition
aANDb = a.read() & b.read();
aXORb = a.read() ^ b.read();
cinANDaXORb = cin.read() & aXORb;

//Calculate sum and carry out
sum = aXORb ^ cin.read();
cout = aANDb | cinANDaXORb;
```

from their technical implementation (see Sect. 3.2), these "idle waiting" times sharply decrease payload performance of the systems. Figure 1b depicts how efficiencies of recent supercomputers depend [6] on the number of single-threaded processors in the system and the parameter $(1-\alpha)$, describing non-payload portion of the corresponding benchmark task. It is known since decades that "*this decay in performance is not a fault of the architecture, but is dictated by the limited parallelism*" [4]; in excessive systems of modern hardware (HW), *is also dictated by laws of nature* [16].

Using shorter operands (half precision rather than double precision) reduces T_A non-proportionally: the housekeeping costs (such as fetching, addressing) remain constant (although the amount of data movement and manipulation decreases). One expects a four-fold performance increase when using half-precision rather than double precision operands [19], and the consumed power consumption data underpin that expectation. However, the measured increase in computing performance was only three times higher: the apparent execution time T_A and the processing time T_p differ.

2.3 Example: Temporal Diagram of a 1-Bit Adder

Although for its end-users, the processor is the "atomic unit" of processing,[7] principles of computing are valid also at "sub-atomic" level of gate operations. Describing the temporal operation at gate level is an excellent example, that *the line-by-line compiling (sequential programming, called also Neumann-style programming [20]), formally introduces only logical dependence, but through its technical implementation it implicitly and inherently introduces a temporal behavior, too.*

The one-bit adder is one of the simplest circuits used in computing. Its common implementation comprises 5 logic gates, 3 input signals and 2 output signals. Gates

[7]The reconfigurable computing, with its customized processors and non-processor-like processing units, does not change significantly the landscape.

are logically connected internally: they provide input and output for each other. The relevant fraction of the equivalent source code is shown in Listing 1.

Figures 2a and b show the timing diagram of a one-bit adder, implemented using common logic gates. The three input signals are aligned on axis y, the five logic gates are aligned on axis x. Gates are ready to operate as well as signals are ready to be processed (at the head of the blue arrows). The logic gates have the same operating time (the length of green vectors), their access time includes the needed multiplexing. Signals must reach their gate (dotted green arrows), that (after its operating time passes) produces its output signal, that starts immediately towards the next gate. Vertical green arrows denote gate processing (one can project the arrow to axis x to find out the ID of the gate), labelled with the name of the produced signal. There are "pointless" arrows in the figure. For example, signal $a\&b$ reaches the OR gate much earlier, than the signal to its other input. Depending on the operands of OR, it may or may not result in the final sum.

Notice, that considering physical distance and finite interaction speed, drastically changes the picture we have (based on "classic computing"), that the operating time of an adder is simply the sum of the corresponding "gate times". For example, the very first AND and XOR operations could work in parallel (at the same time), but the difference in their physical distance the signals must travel, changes the times when they can operate with their signals. Also, compare the temporal behavior of the signal sum on the two figures. The only difference between subfigures is that the second XOR gate moved to another place.

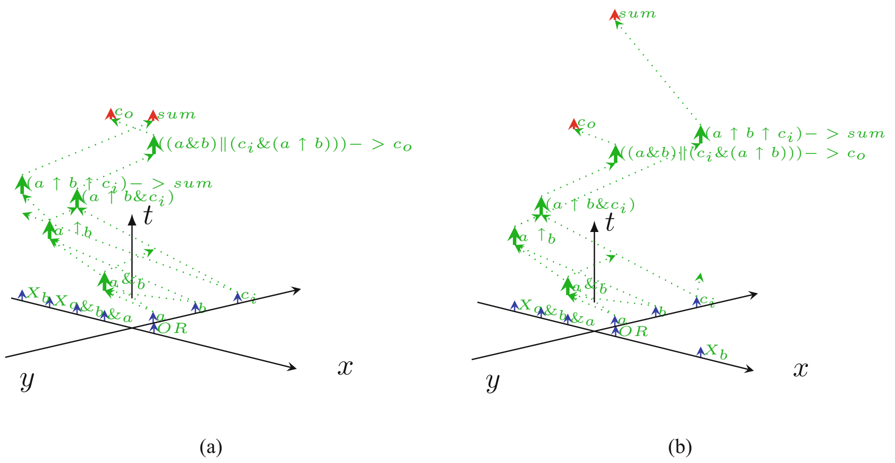


Fig. 2 Temporal diagram of a one-bit adder in time-space system. The diagram shows the logical equivalent of the SystemC source code of Listing 1., *the time from axis x to the bottom of green arrows* signals "idle waiting" time (undefined gate output). Notice how changing position of a gate affects signal timing. (**a**) Temporal dependence diagram of a 1-bit adder. The second XOR gate is at (−1,0) . (**b**) Temporal dependence diagram of a 1-bit adder. The second XOR gate is at (+1,0).

The difference in timing roots not only in the different number of gates involved: the distance traversed by the signals can contribute equally, and even counterbalance the different number of involved gates. As the c_o output is the input c_i for the next bit, is must be wired there. The total execution time of, say, a 64-bit adder shall be optimized at that level, rather than at bit level. Orchestrating temporal operation through considering both complexity of operation, and positions of signals and operators, can significantly enhance performance.

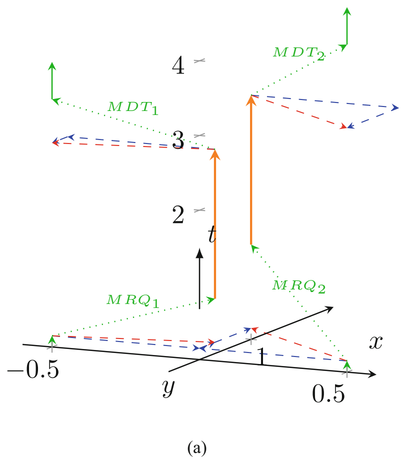
The goal of this section and Figs. 2a and b is only to call the attention to that *in addition to the viewpoint of mathematics (using standard gates and logic functions) and technology (which technology enables to produce smaller gate times and smaller expenses), also the temporal behavior must be considered, when designing chips*. Even inside a simple adder circuit, the performance can be changed significantly, only via changing physical distance of gates; in strong contrast with the "classic computing".

The meaning of "idle waiting" is slightly changed here, *The gates produce valid output only after they received all of their internally-produced operand(s), plus their "gate time", at the head of the corresponding green arrow. The total operating time of the adder is considerably longer than the sum of operating times of its gates.* The proper positioning of gates (and wiring them) is a point to be considered seriously, and maybe also the role of gates must be rethought.

2.4 Using New Effect/Technology/Material in Computing Chain

Given that *apparent processing time T_A* defines performance of the system, T_p (physical processing time, a vector perpendicular to the XY plane) and T_t (transfer time, a vector between different planes) must be concerted. In a complex system, *it is not reasonable to fabricate smaller components without decreasing their processing time proportionally; and similarly, replacing a Processing Unit (PU) with a very much quicker one has only marginal effect, if the physical distance of the PUs cannot be reduced proportionally, at the same time.*

Figure 3 demonstrates why: two different topologies and two different physical cache operating speeds are used in the figure. Two cores are in positions (−0.5,0) and (0.5,0), furthermore two cache memories at (0,0.5) and (0,1). The signal, requesting to access cache, propagates along the dotted green vector (it changes both its time and position coordinates), the cache starts to operate only when the green dotted arrow hits its position. After its operating time (the vertical orange arrow), the result is delivered back to the requesting core. This time can also be projected back to the "position axes", and their sum (red thin arrow) can be calculated. The physical delivery of the fetched value begins at the bottom of the lower vertical green arrows, includes waiting and finishes at the head of the upper vertical green arrows; their distance defines the *apparent cache access time T_A*. Physical cache



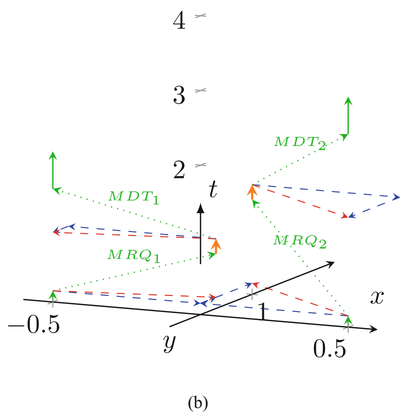


Fig. 3 Performance dependence of an on-chip cache memory, at different cache operating times, in the same topology. Cores at x= −0.5 and x=0.5 positions access on-chip cache at y=0.5 and y=1.0, respectively. Vertical orange arrows represent physical cache operating time, and vertical green arrows the apparent access time. The physical operations speed of cache memory of the right subfigure is 10 times better. Compare the apparent access times to the corresponding physical ones (the time ratio is better only about a factor of two). Notice also that the apparent operating speed is more sensitive to the position rather than to the speed of the cache memory. (**a**) Normal speed cache memory. Two different cache memories, with the same physical cache sped, but at different internal on-chip cache position. (**b**) Super-quick (10 times quicker) cache memory. Assumes new material/physical mechanism. Two different cache memories, with the same physical cache sped, but at different internal on-chip cache position

access time (the vertical orange arrow) begins when signal reaches the cache. Till that time, cache is idle waiting. Core is also idle waiting until the requested content arrives. Notice that *apparent processing time is a monotonic function of the physical processing time, but because of the included—fixed time—'transmission times' due to physical distance of the respective elements, their dependence is far from being linear*. Repeated operation of course can change the idle/to active ratio; one must consider, however, the resources the signal delivery uses.

The apparent processing time (represented by the distance of the vertical green arrows) is only slightly affected by the physical speed of the cache memory (represented by vertical orange arrows). The right subfigure assumes that some new material/technology/effect decreases access time to one tenth of the time assumed on the left subfigure. In the figure, the technology (at considerable expenses) improved physical access time by a factor of ten, but the apparent access speed has improved only by a factor of less than two. *Even if the physical cache time could be reduced to zero, the apparent access time cannot be reduced below the time defined by the respective distances/interaction times. Mimicking the biology is useful also here: the time window, where the decision is made, is of the same size, independently* *of the path traversed by the signal (the axon length) and the speed of the signal (conduction velocity); and is in the order of the 'processing time' of the neurons.*[8]

A recently proposed idea is to replace slow digital processing with quick analog processing [21, 22], and *may be proposed using any future new physical effect and/or material*, such as in [23]: they decrease T_p, but to make them useful for computing, their in-component transmission time T_t, and especially inter-component transmission time must be considerably decreased. *Neglecting their temporal behavior limits the utility of any new method, material or technology, if they are designed/developed/used in the spirit of the old (timeless) paradigm.*

3 Identifying Bottlenecks of Computing Due to Their Technical Implementation

3.1 Synchronous and Asynchronous Operation

The case depicted in Fig.1a is an asynchronous operation: when the light cone arrives at the observer, the second processing can start. If we have additional observers, their T_t^A and T_t^B may be different, and we have no way to synchronize their operation. If we have another observer at the point mirrored to the origo, the light cone arrives at it at about the same T_t^A, but to synchronize the operation of the two observers, we would need $T_{synch} = T_t + T_t^A + T_t^B$. Instead, we issue another light cone (a central clock) at the origo (it the case of that light cone, the processing time is zero, just a rising edge) and observers are instructed to start their processing when this synchronizing light cone reaches their point of observation.

In the *time-space system*, not only observers on the surface of the cone, but also the ones inside the cone, can notice that the first light went on. If T_{synch} is large enough, all observers will notice the first light. After noticing the light, they all can start their processing at that time $t = 2 * T_p + T_{sync}$. Given that both $T_{p,i}$ and $T_{t,i}$ can be different, $T_{synch} \geq T_{p,i} + T_{t,i}$, for any observer i, must be fulfilled. This time is larger than any of the $T_{p,i} + T_{t,i}$ times: *for the rest of observers, the idle time increases*. Given that their internal wiring can be very different, we must choose the clock period according to the "worst-case". *For the rest of observers, this constraint means a significant increase in their value $T_{t,i}$*. All observers must wait for the slowest one. The more observers (and the more steps!), the more waiting. This effect is considerable even inside the chip (at $\leq$ cm distances); in the case of supercomputers, the distance is about 100 m.

A careful analysis [17] discovered that using synchronous computing (using clock signals) has a significant effect on performance of large-scale systems mimicking neuromorphic operation. The performance analysis [25] of large-scale brain

[8]The biology can change the conduction velocity, that needs energy, so finding an optimum is not as simple.

simulation facilities demonstrated an exciting parallel between modern science and large-scale computing. The commonly used 1 ms integration time, limited both the many-thread software (SW) simulator, running on general-purpose supercomputers, and the purpose-build HW brain simulator, to the *same value of payload performance*. Similar shall be the case very soon in connection with building the targeted large-scale neuromorphic systems, despite the initial success of specialized neuronal chips (such as [26, 27]). Although at a higher value (about two orders of magnitude higher than the one in [25]), systems built from such chips also shall stall because of the "*quantal nature of time*" [16], although using asynchronous operating mode can slinghtly rearrange the scene.

3.2 The High Speed Serial Bus

Components of technical computing systems (including biology-mimicking neuromorphic ones) are connected through a set of wires, called "bus". The bus is essentially the physical appearance of the "technical implementation" of communication, stemming from the SPA, as illustrated in Fig. 4. The inset shows a simple neuromorphic use case: one input neuron and one output neuron communicating through a hidden layer, comprising only two neurons. Figure 4a mostly shows *the biological implementation: all neurons are directly wired to their partners*, i.e., a system of "parallel buses" (axons) exists. Notice that the operating time also comprises two non-payload times (T_t): data input and data output, which coincide with the non-payload time of the other communication party. The diagram displays logical and temporal dependencies of the neuronal functionality. The payload operation ("the computing") can only start after data is delivered (by the, from this point of view, non-payload functionality: input-side communication), and output communication can only begin when the computing finished. Importantly, *communication and calculation mutually block each other*. Two important points that neuromorphic systems must mimic noticed immediately: i/ *the communication time is an integral part of the total execution time*, and ii/ *the ability to communicate is a native functionality* of the system. In such a parallel implementation, *performance of the system*, measured as the resulting total time (processing + transmitting), *scales linearly with increasing both non-payload communication speed and payload processing speed.*

Figure 4b shows a *technical implementation of a high-speed shared bus* for communication. To the right of the grid, the activity that loads the bus at the given time is shown. A double arrow illustrates communication bandwidth, the length of which is proportional to the number of packages the bus can deliver in a given time unit. We assume that the input neuron can send its information in a single message to the hidden layer, furthermore, that the processing by neurons in the hidden layer both starts and ends at the same time. However, the neurons must compete for accessing the bus, and only one of them can send its message immediately, the other(s) must wait until the bus gets released. The output neuron can only receive the

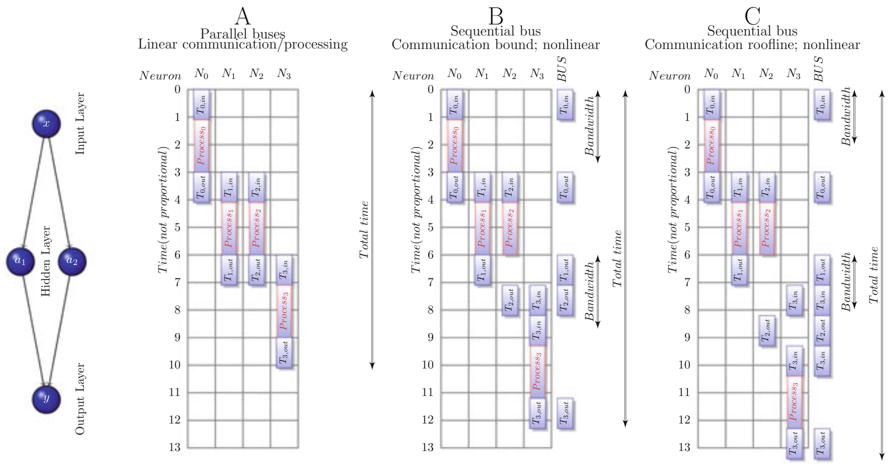


Fig. 4 Implementing neuronal communication in different technical approaches. (**a**): the parallel bus; (**b**) and (**c**): the shared serial bus, before and after reaching the communication "roofline" [24]

message when the first neuron completed it. Furthermore, the output neuron must first acquire the second message from the bus, and the processing can only begin after having both input arguments. *This constraint results in sequential bus delays both during non-payload processing in the hidden layer and payload processing in the output neuron.* Adding one more neuron to the layer, introduces one more delay.

Using the formalism introduced above, $T_t = 2 \cdot T_B + T_d + X$, i.e., the bus must be reached in time T_B (not only the operand delivered to the bus, but also waiting for arbitration: the right to use the bus), twice, plus the physical delivery through the bus. The X denotes "foreign contribution": if the bus is not dedicated for "neurons in this layer only", any other traffic also loads the bus: both messages from different layers and the general system messages may make processing slower.

Even if only one single neuron exists in the hidden layer, it must use the mechanisms of sharing the bus, case by case. The physical delivery to the bus takes more time than a transfer to a neighboring neuron (both the arbiter and the bus are in *cm* distance range). If we have more neurons (such as a hidden layer) on the bus, and they work in parallel, they all must wait for the bus. The high-speed bus is very slightly loaded when only a couple of neurons are present, and its load increases linearly with the number of neurons in the hidden layer (or, maybe, all neurons in the system). The temporal behavior of the bus, however, is different. Under the biology-mimicking workload, the second neuron must wait for all its inputs originating in the hidden layer. If we have L neurons in the hidden layer, the transmission time of the neuron behind the hidden layer is $T_t = L \cdot 2 \cdot T_B + T_d + X$. This temporal behavior explains why "*shallow networks with many neurons per layer … scale worse than deep networks with less neurons*" [10]: *the physical bus delivery time T_d, as well as the processing time T_p, become marginal if the layer* *forces to make many arbitrations to reach the bus*: the number of the neurons in the hidden layer defines the transfer time (Recall Fig. 1a for the consequences of increasing the transfer time). In deeper networks, the system sends its messages at different times in different layers (and, even they may have independent buses between the layers), although *the shared bus persists in limiting the communication*. Notice that there is no way to organize the message traffic: only one bus exists.

Figure 5 discusses, in terms of "temporal logic", the case depicted in the inset in Fig. 4 (where the same operation is discussed in conventional terms): why using high-speed buses for connecting modern computer components leads to very severe performance loss, especially when one attempts to imitate neuromorhic operation. The two neurons of the hidden layer are positioned at (−0.3,0) and (0.6,0). The bus is at position (0,0.5). The two neurons make their computation (green arrows at the position of neurons), then they want to tell their result to fellow neurons. Unlike in biology, first they must have access to the shared bus (red arrows). Core at (−.3,0) is closer to the bus, so its request is granted. As soon as the grant signal reaches requesting core, the bus operation is initiated, and the data starts to travel to the bus. As soon as it reaches the bus, it is forwarded by the high speed of the bus, and at that point bus request of the other core is granted, and finally, also calculated result of the second neuron is bused.

At this point comes into picture the role of the workload on the system: the two neurons in the hidden layer want to use the single shared bus, at the same time, for communication. As a consequence, *the apparent processing time is several times higher, than the physical processing time, and it increases linearly with the number of neurons in the hidden layer* (and maybe with also the total number of neurons in the system, if a single high-speed bus is used). *In vast systems, especially when attempting to mimic neuromorphic workload, the speed of the bus is getting marginal*. Notice that times shown in the figure are not proportional: the (temporal) distances between cores are in the several picoseconds range, while the bus (and the arbiter) are at a distance well above nanoseconds, so *the actual temporal behavior (and the idle time stemming from it) is much worse than the figure suggests*. This

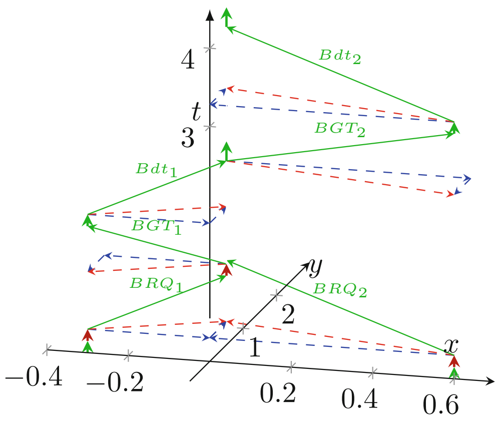
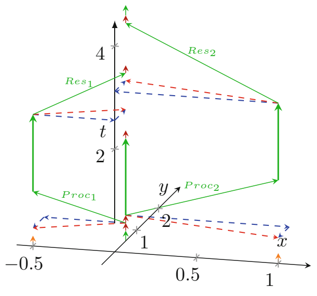


Fig. 5 The operation of the sequential bus, in time-space coordinate system system. Near to axis *t*, *the lack of vertical arrows* signals "idle waiting" time

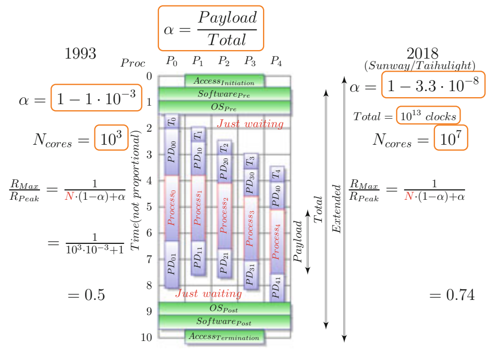
is why "*The idea of using the popular shared bus to implement the communication medium is no longer acceptable, mainly due to its high contention*" [28]. The figure suggests to use another design principle instead of using the bus exclusively, directly from the position of the computing component (Fig. 5).

Given that the bus bandwidth is finite, there comes the point when the amount of messages exceeds the available bus bandwidth. Figure 4c demonstrates the case, where for better visibility, the bus bandwidth is lower, but the required packet bandwidth slice is the same. In this case, the second neuron in the hidden layer cannot send its message when the first one finishes its transmission: the bus transmission roofline [24] is reached. In that case the transmission time shall be extended with a new term $T_t = (B + L) \cdot 2 \cdot T_B + T_d + X$, where B is the number of messages above the number of messages that the bus can deliver in a unit time. Reaching the roofline causes further extra delay in both non-payload and payload processing times, extending the total execution time. A single sequential bus can deliver messages only one after the other, i.e., *increasing number of neurons increases utilization of the bus and prolongs total execution time as well as apparent processing time of the individual neurons*. This effect can be so strong in large systems, that emergency measures must have been introduced: the events "*are processed as they come in and are dropped if the receiving process is busy over several delivery cycles*" [25].

When using a shared bus, increasing either processing speed or communication speed does not affect linearly the total execution time any more. Furthermore, it is not the bus speed that limits performance. Recall Fig. 1a again, to see, how the time projection of a relatively small increase in the transfer time T_t can lead to a relatively large change in the value of apparent processing time T_A; and so



(a)



(b)

Fig. 6 The parallelized sequential operation as described in the proposed time-space system, with its simplified, non-technical model [6]. (**a**) Time diagram of parallelized sequential operation in time-space. (**a**) A non-technical, simplified model of parallelized sequential computing operations. Notice the different nature of those contributionsand that they have only one common feature: *they all consume time*. The vertical scale displays the actual activity for processing units shown on the horizontal scale

leads to incomprehensible slowdown of the system: the slowest component defines efficiency. Conventional way of communication may work fine as long as there is no competition for the bus, but leads to queuing of messages in the case of (more than one!) independent sources of communication. The bursty nature, caused by the need of central synchronization, tops the effect, and leads to a "communicational collapse" [29], that denies huge many-processor systems, especially neuromorphic ones [30].

To have a chance to connect a large number of computing units in biology-mimicking systems, drastically new bus system and drastically new traffic organization is required [31]. *Using a single high-speed bus greatly contributes to the experienced very low efficiency of Artificial Neural Network (ANN)s* [5], *and finally that "Core progress in AI has stalled in some fields"* [32].

3.3 Parallelized Sequential Processing

Present technical approaches assume a linear dependence between payload and nominal performances of computing systems as "*Gustafson's formulation [7] gives an illusion that as if N [the number of the processors] can increase indefinitely*" [33]. The fact that "*in practice, for several applications, the fraction of the serial part happens to be very, very small thus leading to* ***near-linear*** *speedups*" [33] (see also value of α in Fig. 1b), however, misled the researchers. *Gustafson's "linear scaling" neglects all non-payload contributions entirely, including the temporal behavior of the components*. He established his conclusions on only several hundred processors. The interplay of improving parallelization and general HW development (including non-determinism of modern HW [34]) covered for decades that *weak scaling was used far outside of its range of validity* [5]. In our terminology, Gustafson's assumption means that $T_t = 0$, which is not the case, in any computing system, and especially not in the case of neuromorphic computing systems. As pointed out above, having idle time in computing systems is inevitable; the vastly increased number of idle cycles due to physical size and operating mode of computing systems led to the effects detailed above.

Figure 6a depicts temporal diagram of distributed parallel processing in the introduced time-space system. One of the PUs (in our case the one at (0,0.5)) orchestrates the operation, including receiving the start command and returning the result. This core makes some sequential operations (such as initializing data structures, short green arrow), then it sends the start command and operands to fellow cores at (−0.5,0) and (1,0), one after the other. Signal propagation takes time (depending on distance from the coordinator), and after that time, fellow cores can start their calculation (their part of the parallelized portion). Of course, orchestrator PU must start all fellow PUs (red arrows), then it can start its portion of distributed processing. In the case of large number of fellow processors, it may be advantageous if the coordinator does not have its own portion of parallelizable code:[9] executing that code may delay receiving results from the fellow processors.

As fellow PUs finish their portion, they must transmit their data Res_i to the orchestrator, that receives those data in *sequential* mode, and finally makes some closing *sequential* processing. Again, the *inherently sequential-only portion* [35] of the task increases with number of cores and its *idle waiting time* (time delay of signals) increases with physical size (cable length). The times shown in the figure are not proportional, and largely depend on type of the system. For example, in supercomputers, total calculation time is in the hours range, number of red arrows (without clustering) can be up to several millions, and the delay, due to the finite speed of signal propagation, in several thousand clock cycles (in the case of using Ethernet networks, several millions).

Notice, that the figures assume no dependence (such as logical dependence on sharing physical resources) between the computing objects (threads), and especially not the case when several SW threads share the same PU. Notice also, that the orchestrating PU must wait results from all fellow PUs, i.e. *the slowest branch defines performance*.

Amdahl listed [36] different reasons why losses in "computational load" can occur. Fortunately, Amdahl's idea enables us to *put everything that cannot be parallelized, i.e., distributed between the fellow processing units, into the sequential-only fraction*. For describing the parallel operation of sequentially working units, the model depicted in Fig. 6b was prepared. The technical implementations of the different parallelization methods show up virtually infinite variety [37], so we present here a (by intention) strongly simplified, non-technical, model. The model has some obvious limitations, among others, because of the non-determinism of modern HW systems [34, 38].

In addition to "idle time"s discussed above, the serialized parallel processing adds one more contribution. Even the simplest (parallelized sequential) task has a non-parallelizable *portion of time*, that—according to Amdahl's Law—limits the achievable payload computing performance. Here the *sequential bus* and the *transmission delay* play a role, again. Because, in the SPA, the initiating processor can address only one processor (or through clustering: only a few of them), the other processors must make additional *idle waiting*: the loop to address them takes time, and the cable length significantly increases their T_t. This effect, however, comes to light only at a relatively high number of cores *and* real-life workloads. At a lower number of cores *and* HPL-class benchmarks, only a slight deviation from the linearity, predicted by the "weak scaling", can be noticed.

The right subfigure in Fig. 7 displays the payload performance of a many-processor SPA system when executing different workloads (that define the non-payload to payload ratio); for the math details see [6, 16]. The top diagram lines represent the best payload performance that the supercomputers can achieve when running the benchmark HPL, which represents the minimum communication

[9]For examples see the architectures of supercomputers *Taihulight* and *Summit*.

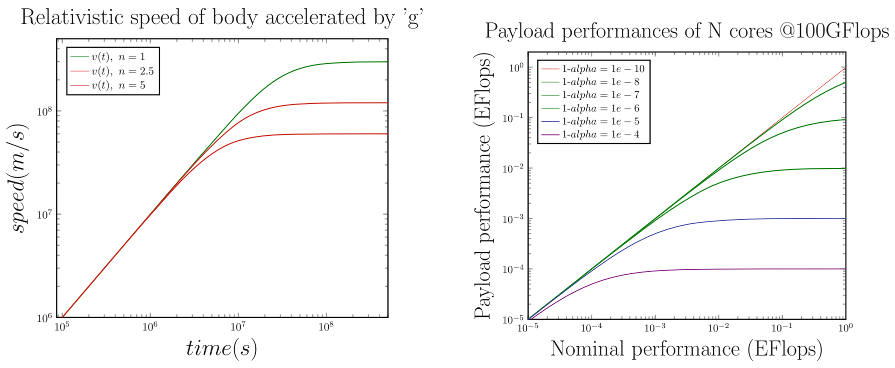


Fig. 7 The limiting effect considered in the "modern" theories. One left side, the speed limit, as explained by the theory of relativity, is illustrated. The refractory index of the medium defines the value of the speed limit. On the right side, the payload performance limit of the parallelized sequential computing systems, as explained by the "modern paradigm", is illustrated. The ratio of the non-payload to payload processing defines the value of the payload performance

a parallelized sequential system needs. *The bottom diagram line represents the estimation of the payload performance that neuromorphic-type processing can achieve in SPA systems (See also Fig. 1b).* Notice the similarity with the left subfigure: *under extreme conditions, in the science, an environment-dependent speed limit exist, and in computing, a workload-dependent payload performance limit exists* [16].

To have hopes to significantly increase computing performance of our present cutting-edge conventional and future neuromorphic computing systems, principle other than parallelizing otherwise sequentially processing systems, must be discovered. The recent paradigm leads to not only inherent performance limits, but also to irrationally high power consumption.

3.4 Communication

One of the worst computing performance limiting factors is the method of communication between processors, which increases exponentially with increasing complexity/number. Historically, in the model of computing proposed by von Neumann, there was one single entity, an isolated (non-communicating) processor, whereas in bio-inspired models, billions of entities, organized into specific assemblies, cooperate via communication. (Communication here means not only sending data, but also sending/receiving signals, including synchronization of the operation of entities.) Neuromorphic systems, expected to perform tasks in one paradigm, but assembled from components manufactured using principles of (and implemented by experts trained in) the other paradigm, are unable to perform at the required speed and efficacy for real-world solutions. The larger the system, the higher the communication load and the performance debt. With reference to Fig. 1a, *time contribution of the communication is part of the processing time T_p*, although the overwhelming part of it could be done in parallel with the computing activity. This feature both decreases available processing capacity of a neuron, and strongly changes value of R. More importantly, it must use communication facilities through Input/Output (I/O) instructions, wasting a massive amount of time for that.

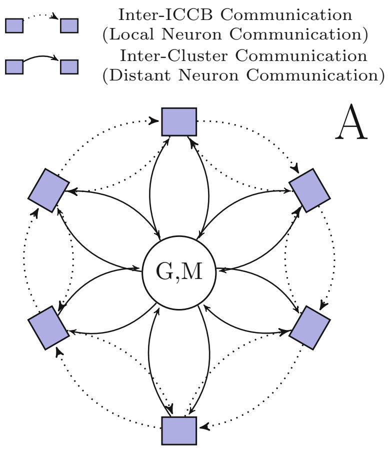
4 The Effect of Temporal Behavior on Scaling

Dependence of *payload* performance on *nominal* performance in many-many processor systems is strongly nonlinear at higher performance values (implemented using a large number of processors). This effect is especially disadvantageous for networks, such as neuromorphic ones, that show up non-proportionally much idle wait time, mainly because of the reasons presented above. The linear dependence at low nominal performance values explains why initial successes of *any new technology, material or method* in the field, using the classic computing model, can be misleading: in simple cases classic paradigm performs tolerably well thanks to that *compared to biological neural networks, current neuron/dendrite models are simple, the networks small and learning models appear to be rather basic*.

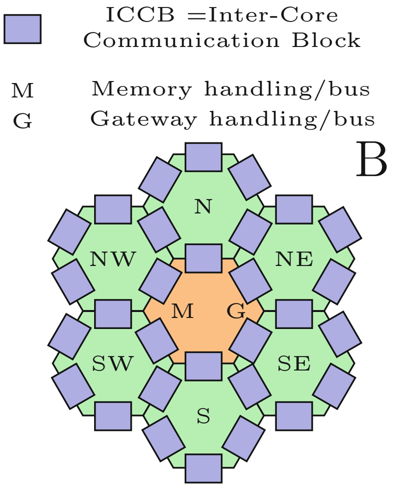
The biology is aware of that the transmission time is a crucial part of the processing. "*Importantly, distally projecting axons of long-range interneurons have several-fold thicker axons and larger diameter myelin sheaths than do pyramidal cells, allowing for considerably faster axon conduction velocity*" [39]. Faster conduction increases the energy consumption of a cell (needing more myelin), but it prevents a race condition between the signals. The biology "wastes" extra energy only when required, and here there appears the need to refine the "fire and wire together" operating principle with modulating the conduction velocity. The surprising resemblance between Fig. 8a and Fig. 7 in [39] also underlines the importance of making a clear distinction between handling 'near' and 'far' signals. Although the Inter-Core Communication Block (ICCB) blocks in the biology-mimicking architecture [31] can adequately represent 'locally connected' interneurons and the 'G' gateway the 'long-range interneurons', the biological conduction time must be separately maintained. Computer technology cannot speed up communication selectively, as biology does, and it is not worth to slow it down selectively. Making time-stamps and relying on computer network delivery principles is not sufficient: *temporal behavior is a vital feature of biology-mimicking systems and we must not replace them with synchronization principles of computing*.

5 Summary

Statements such as "*The von Neumann architecture* is fundamentally inefficient and non-scalable for representing massively interconnected neural networks" [40]


Inter-ICCB Communication
(Local Neuron Communication)
Inter-Cluster Communication
(Distant Neuron Communication)
A
G,M

(a)



(b)

Fig. 8 The communication scheme between local and farther neurons, as can be implemented in technically [31]. (**a**) The conceptual communication diagram (compare to Fig. 7 in [39]), mimicking the communication between local neurons the farther neurons. (**b**) The proposed implementation: the Inter-Core Communication Blocks represent a "local bus" (directly wired, with no contention), while the cores can communicate with the cores in other clusters through the 'G' gateway as well as the 'M' (local and global) memory

should be modified like this "*the architectures based on the non-temporal abstraction proposed by von Neumann*". Especially the figures above, provide a very clear pointer: *to make efficient and large systems (including neuromorphic ones), the fundamental principles of operation of computing, communication, including the bus system and principle of handling messages, as well as the cooperation between processors, must be scrutinized and drastically changed.* Comprehending the timely behavior of the components can serve as a good starting point to do so.

References

1. K. Asanovic, R. Bodik, J. Demmel, T. Keaveny, K. Keutzer, J. Kubiatowicz, N. Morgan, D. Patterson, K. Sen, J. Wawrzynek, D. Wessel, K. Yelick, A view of the parallel computing landscape. Commun. ACM **52**(10), 56–67 (2009)
2. US National Research Council, The Future of Computing Performance: Game Over or Next Level? (2011). [Online]. Available: http://science.energy.gov/~/media/ascr/ascac/pdf/meetings/mar11/Yelick.pdf
3. I. Markov, Limits on fundamental limits to computation. Nature **512**(7513), 147–154 (2014)
4. J.P. Singh, J.L. Hennessy, A. Gupta, Scaling parallel programs for multiprocessors: Methodology and examples. Computer **26**(7), 42–50 (1993)

5. J. Végh, Which scaling rule applies to Artificial Neural Networks, in *Computational Intelligence (CSCE) The 22nd Int'l Conf on Artificial Intelligence (ICAI'20)* (IEEE, 2020). Accepted ICA2246, in print. [Online]. Available: http://arxiv.org/abs/2005.08942
6. J. Végh, Finally, how many efficiencies the supercomputers have? J. Supercomput. (2020). [Online]. Available: https://doi.org/10.1007%2Fs11227-020-03210-4
7. J.L. Gustafson, Reevaluating Amdahl's law. Commun. ACM **31**(5), 532–533 (1988)
8. C. Liu, G. Bellec, B. Vogginger, D. Kappel, J. Partzsch, F. Neumäker, S. Höppner, W. Maass, S.B. Furber, R. Legenstein, C.G. Mayr, Memory-efficient deep learning on a SpiNNaker 2 prototype. Frontiers Neurosci. **12**, 840 (2018). [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2018.00840
9. Top500.org, Retooled Aurora Supercomputer Will Be America's First Exascale System (2017). https://www.top500.org/news/retooled-aurora-supercomputer-will-be-americas-first-exascale-system/
10. J. Keuper, F.-J. Preundt, Distributed training of deep neural networks: Theoretical and practical limits of parallel scalability, in *2nd Workshop on Machine Learning in HPC Environments (MLHPC)* (IEEE, 2016), pp. 1469–1476. [Online]. Available: https://www.researchgate.net/publication/308457837
11. J. Végh, How deep the machine learning can be, ser. *A Closer Look at Convolutional Neural Networks* (Nova, In press, 2020), pp. 141–169. [Online]. Available: https://arxiv.org/abs/2005.00872
12. US DOE Office of Science, Report of a Roundtable Convened to Consider Neuromorphic Computing Basic Research Needs (2015). https://science.osti.gov/-/media/ascr/pdf/programdocuments/docs/Neuromorphic-Computing-Report_FNLBLP.pdf
13. G. Bell, D.H. Bailey, J. Dongarra, A.H. Karp, K. Walsh, A look back on 30 years of the Gordon Bell Prize. Int. J. High Performance Comput. Appl. **31**(6), 469–484 (2017). [Online]. Available: https://doi.org/10.1177/1094342017738610
14. S(o)OS project, Resource-independent execution support on exa-scale systems (2010). http://www.soos-project.eu/index.php/related-initiatives
15. Machine Intelligence Research Institute, Erik DeBenedictis on supercomputing (2014). [Online]. Available: https://intelligence.org/2014/04/03/erik-debenedictis/
16. J. Végh, A. Tisan, The need for modern computing paradigm: Science applied to computing, in *Computational Science and Computational Intelligence CSCI The 25th Int'l Conf on Parallel and Distributed Processing Techniques and Applications* (IEEE, 2019), pp. 1523–1532. [Online]. Available: http://arxiv.org/abs/1908.02651
17. J. Végh, How Amdahl's Law limits the performance of large artificial neural networks. Brain Informatics **6**, 1–11 (2019). [Online]. Available: https://braininformatics.springeropen.com/articles/10.1186/s40708-019-0097-2/metrics
18. R. Hameed, W. Qadeer, M. Wachs, O. Azizi, A. Solomatnikov, B.C. Lee, S. Richardson, C. Kozyrakis, M. Horowitz, Understanding sources of inefficiency in general-purpose chips, in *Proceedings of the 37th Annual International Symposium on Computer Architecture*, ser. ISCA '10 (ACM, New York, NY, USA, 2010), pp. 37–47. [Online]. Available: http://doi.acm.org/10.1145/1815961.1815968
19. A. Haidar, P. Wu, S. Tomov, J. Dongarra, Investigating half precision arithmetic to accelerate dense linear system solvers, in *Proceedings of the 8th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems*, ser. ScalA '17 (ACM, New York, NY, USA, 2017), pp. 10:1–10:8
20. J. Backus, Can programming languages be liberated from the von Neumann Style? A functional style and its algebra of programs. Commun. ACM **21**, 613–641 (1978)
21. E. Chicca, G. Indiveri, A recipe for creating ideal hybrid memristive-CMOS neuromorphic processing systems. Appl. Phys. Lett. **116**(12), 120501 (2020). [Online]. Available: https://doi.org/10.1063/1.5142089
22. Building brain-inspired computing. Nature Communications **10**(12), 4838 (2019). [Online]. Available: https://doi.org/10.1038/s41467-019-12521-x

23. P. Cadareanu, et al., Rebooting our computing models, in *Proceedings of the 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)* (IEEE Press, 2019), pp. 1469–1476
24. S. Williams, A. Waterman, D. Patterson, Roofline: An insightful visual performance model for multicore architectures. Commun. ACM **52**(4), 65–76 (2009)
25. S.J. van Albada, A.G. Rowley, J. Senk, M. Hopkins, M. Schmidt, A.B. Stokes, D.R. Lester, M. Diesmann, S.B. Furber, Performance comparison of the digital neuromorphic hardware SpiNNaker and the neural network simulation software NEST for a full-scale cortical microcircuit model. Frontiers Neurosci. **12**, 291 (2018)
26. F. Akopyan, Design and tool flow of IBM's TrueNorth: An ultra-low power programmable neurosynaptic chip with 1 million neurons, in *Proceedings of the 2016 on International Symposium on Physical Design*, ser. ISPD '16 (ACM, New York, NY, USA, 2016), pp. 59–60. [Online]. Available: http://doi.acm.org/10.1145/2872334.2878629
27. M. Davies, et al, Loihi: A neuromorphic manycore processor with on-chip learning. IEEE Micro **38**, 82–99 (2018)
28. L. de Macedo Mourelle, N. Nedjah, F.G. Pessanha, *Reconfigurable and Adaptive Computing: Theory and Applications*, ch. 5: Interprocess Communication via Crossbar for Shared Memory Systems-on-chip (CRC press, 2016)
29. S. Moradi, R. Manohar, The impact of on-chip communication on memory technologies for neuromorphic systems. J. Phys. D Appl. Phys. **52**(1), 014003 (2018)
30. S.B. Furber, D.R. Lester, L.A. Plana, J.D. Garside, E. Painkras, S. Temple, A.D. Brown, Overview of the SpiNNaker system architecture. IEEE Trans. Comput. **62**(12), 2454–2467 (2013)
31. J. Végh, How to extend the Single-Processor Paradigm to the Explicitly Many-Processor Approach, in *2020 CSCE, Fundamentals of Computing Science* (IEEE, 2020). Accepted FCS2243, in print. [Online]. Available: https://arxiv.org/abs/2006.00532
32. M. Hutson, Core progress in AI has stalled in some fields. Science **368**, 6494/927 (2020)
33. Y. Shi, Reevaluating Amdahl's Law and Gustafson's Law (1996). https://www.researchgate.net/publication/228367369_Reevaluating_Amdahl's_law_and_Gustafson's_law
34. V. Weaver, D. Terpstra, S. Moore, Non-determinism and overcount on modern hardware performance counter implementations, in *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 215–224 (April 2013)
35. F. Ellen, D. Hendler, N. Shavit, On the inherent sequentiality of concurrent objects. SIAM J. Comput. **43**(3), 519–536 (2012)
36. G.M. Amdahl, Validity of the single processor approach to achieving large-scale computing capabilities," in *AFIPS Conference Proceedings*, vol. 30, pp. 483–485 (1967)
37. K. Hwang, N. Jotwani, *Advanced Computer Architecture: Parallelism, Scalability, Programmability*, 3rd edn. (McGraw Hill, 2016)
38. P. Molnár, J. Végh, Measuring performance of processor instructions and operating system services in soft processor based systems, in *18th Internat. Carpathian Control Conf. ICCC*, pp. 381–387 (2017)
39. G. Buzsáki, X.-J. Wang, Mechanisms of gamma oscillations. Ann. Rev. Neurosci. **3**(4), 19:1–19:29 (2012)
40. J. Sawada et al., TrueNorth ecosystem for brain-inspired computing: Scalable systems, software, and applications, in *SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 130–141 (2016)