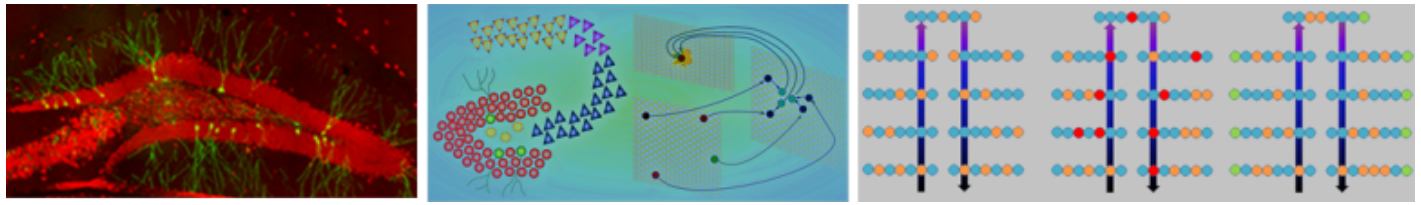
SAND2021-2843C

Sandia National Laboratories

A Neuromorphic Future for Classic Computing Tasks



PRESENTED BY

Brad Aimone

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Article | Published: 23 October 2019

Quantum supremacy using a programmable superconducting processor

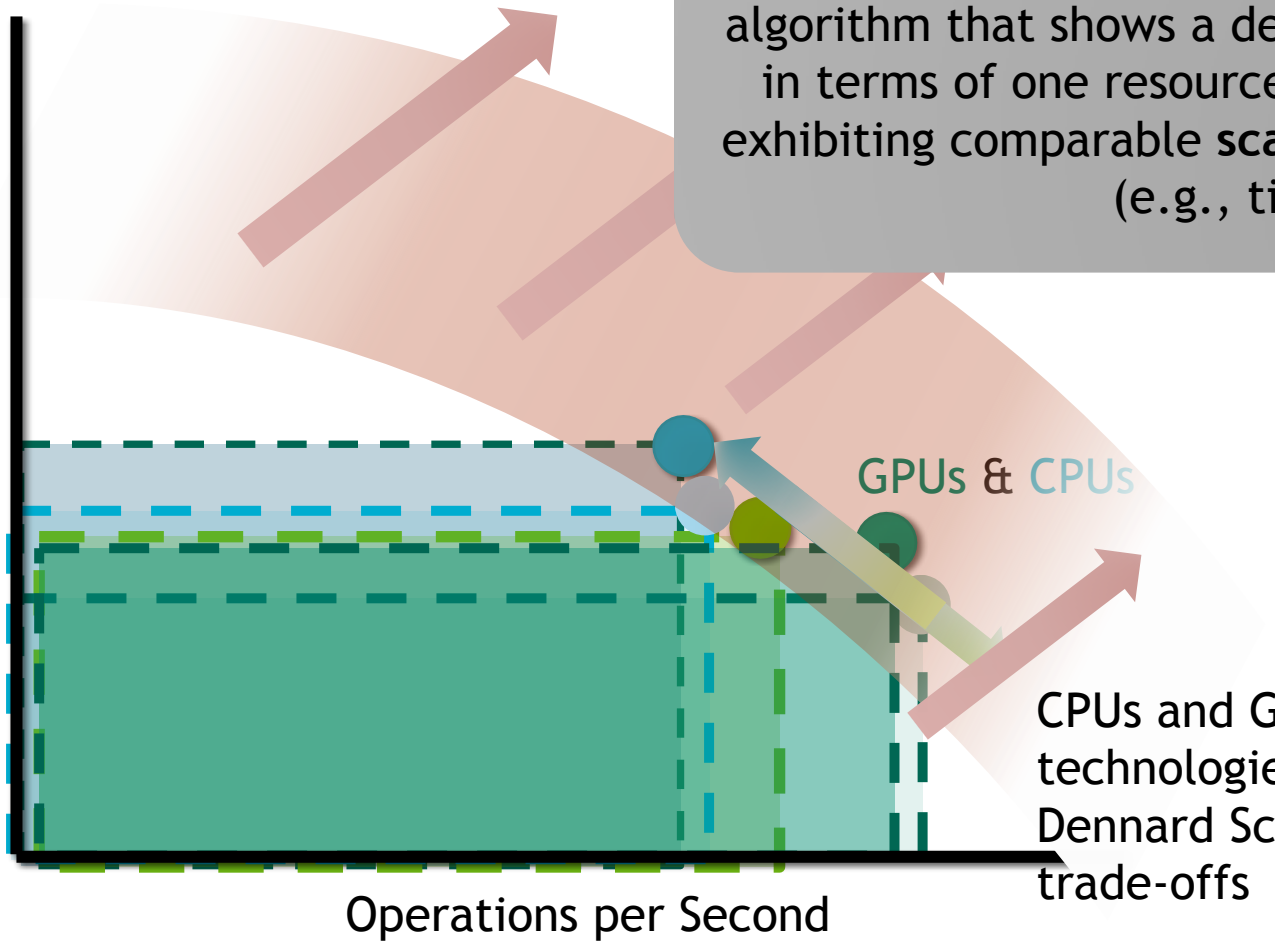
Frank Arute, Kunal Arya, [...] John M. Martinis

Nature **574**, 505–510(2019) | Cite this article

Can we point to a clear computational advantage for neuromorphic computing?

What is a neuromorphic advantage?

We define a *neuromorphic advantage* as an algorithm that shows a demonstrable **advantage** in terms of one resource (e.g., energy) while exhibiting comparable **scaling** in other resources (e.g., time).

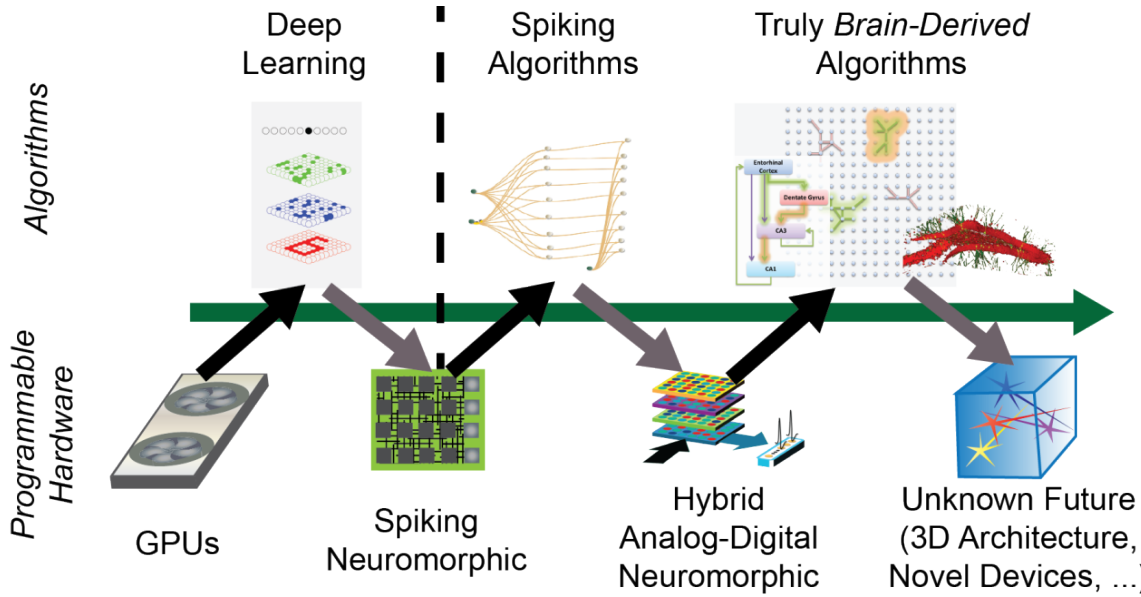


CPUs and GPUs are mature technologies, so post-Moore's Law / Dennard Scaling we expect bounded trade-offs

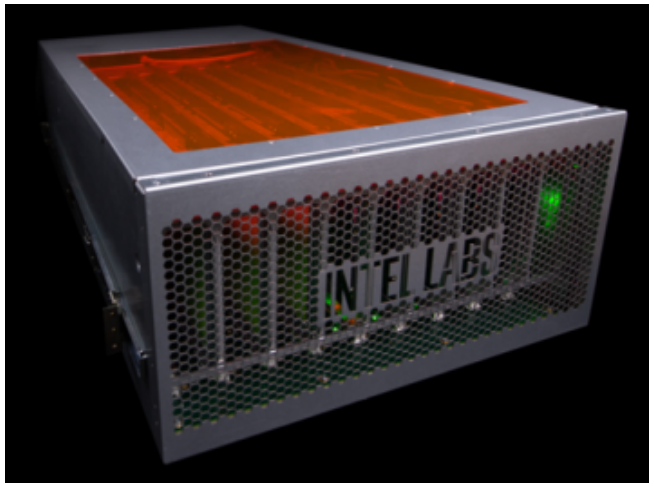
Smith et al., in review 2021

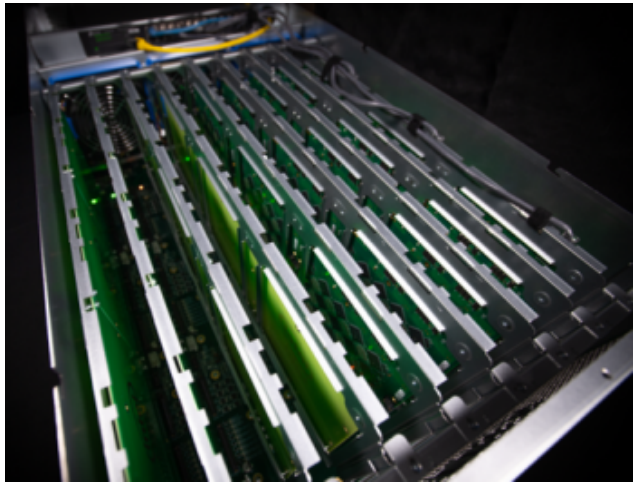
Cutting through the noise: state of neuromorphic computing *today*

- Area of long-term research *and* near-term capability
  - Future may look different (analog, optical, native learning algorithms, self-reconfigurability, etc.)
  - Today's technologies are a *functional* step towards this long-term promise
    - More 1990s GPUs as opposed to today's quantum computers
- Spiking CMOS neuromorphic systems are commercially feasible *today*
  - Standard fabrication technology with established chip design processes
  - Deterministic with pseudo-random capabilities
  - Low power primarily from event-driven computation
  - Arbitrarily scalable
  - Fully programmable from neural perspective



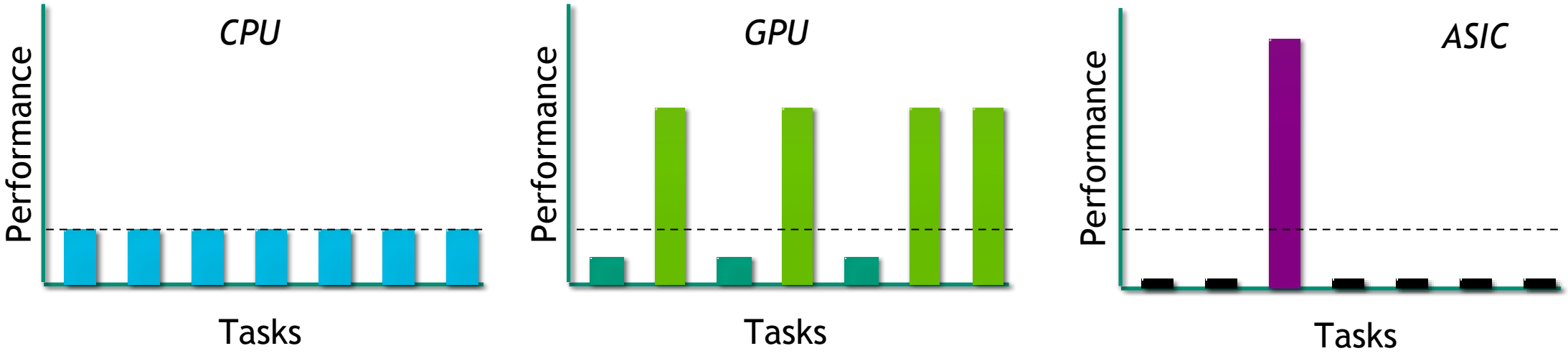
Aimone et al., Advanced Intelligence Systems, 2020





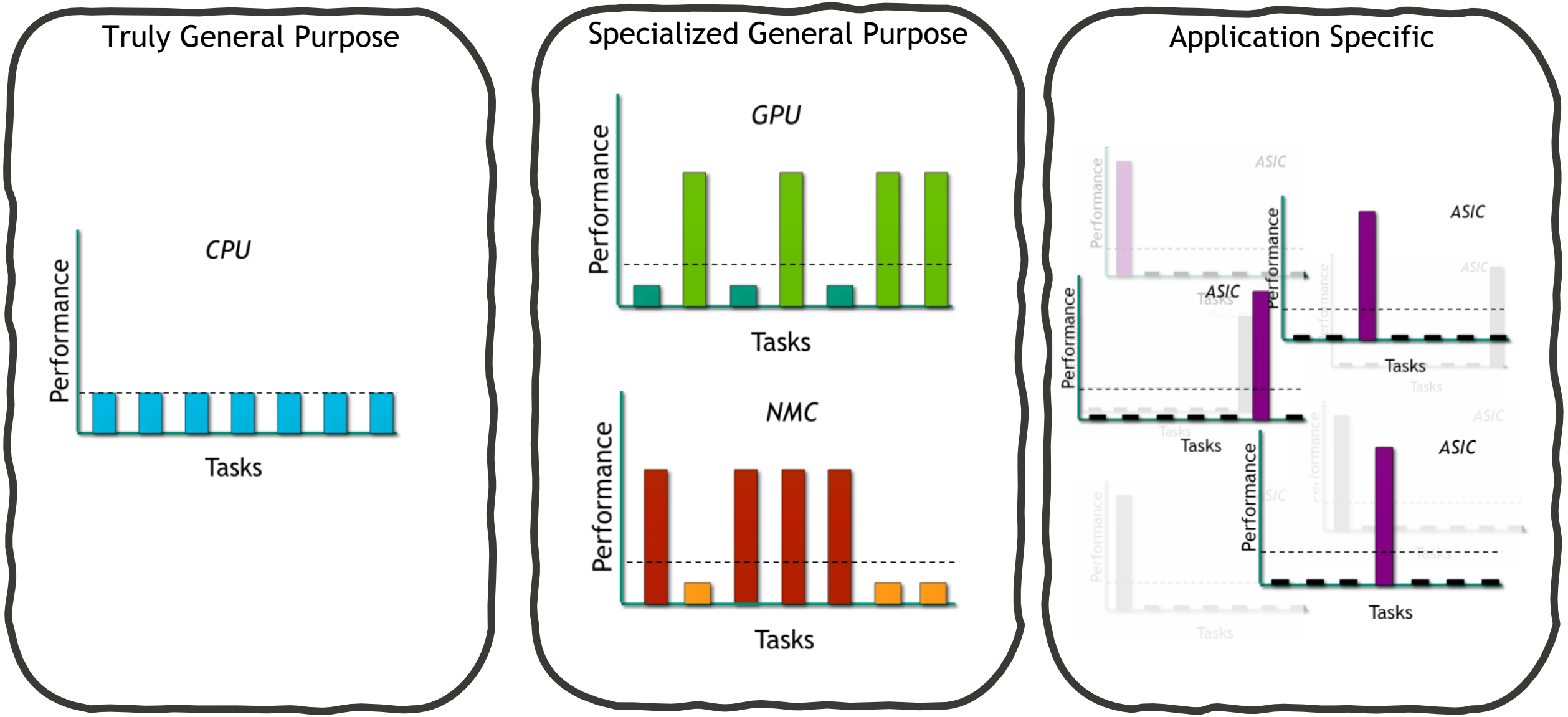
Neuromorphic and the “No Free Lunch” Theorem

No Free Lunch (Wolpert & Macready, 1997) Theorem implies that optimizations to improve some calculations will make other calculations more expensive

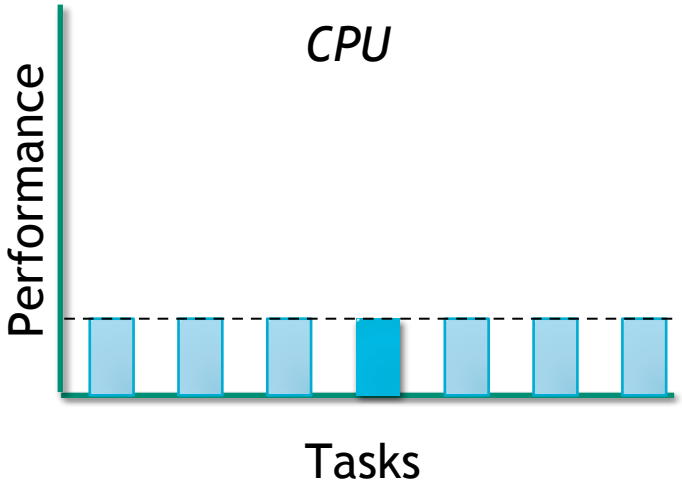


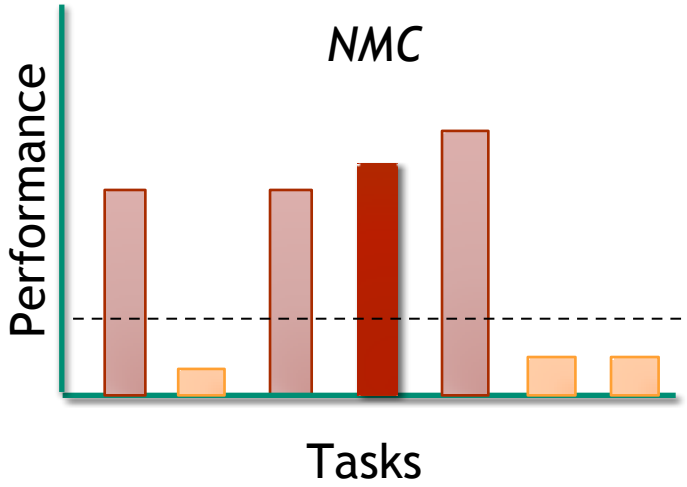
Customizing architecture to be good at one thing makes it worse at everything else

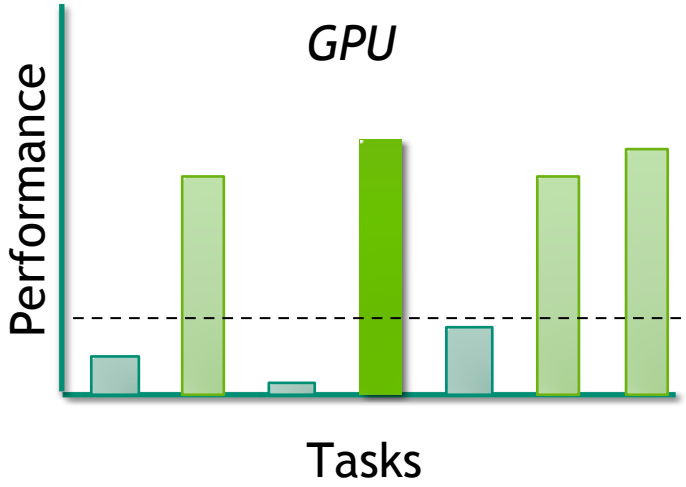
Neuromorphic is similar to GPUs in degree of specialization



The problem with deep learning and NMC's "killer app"…

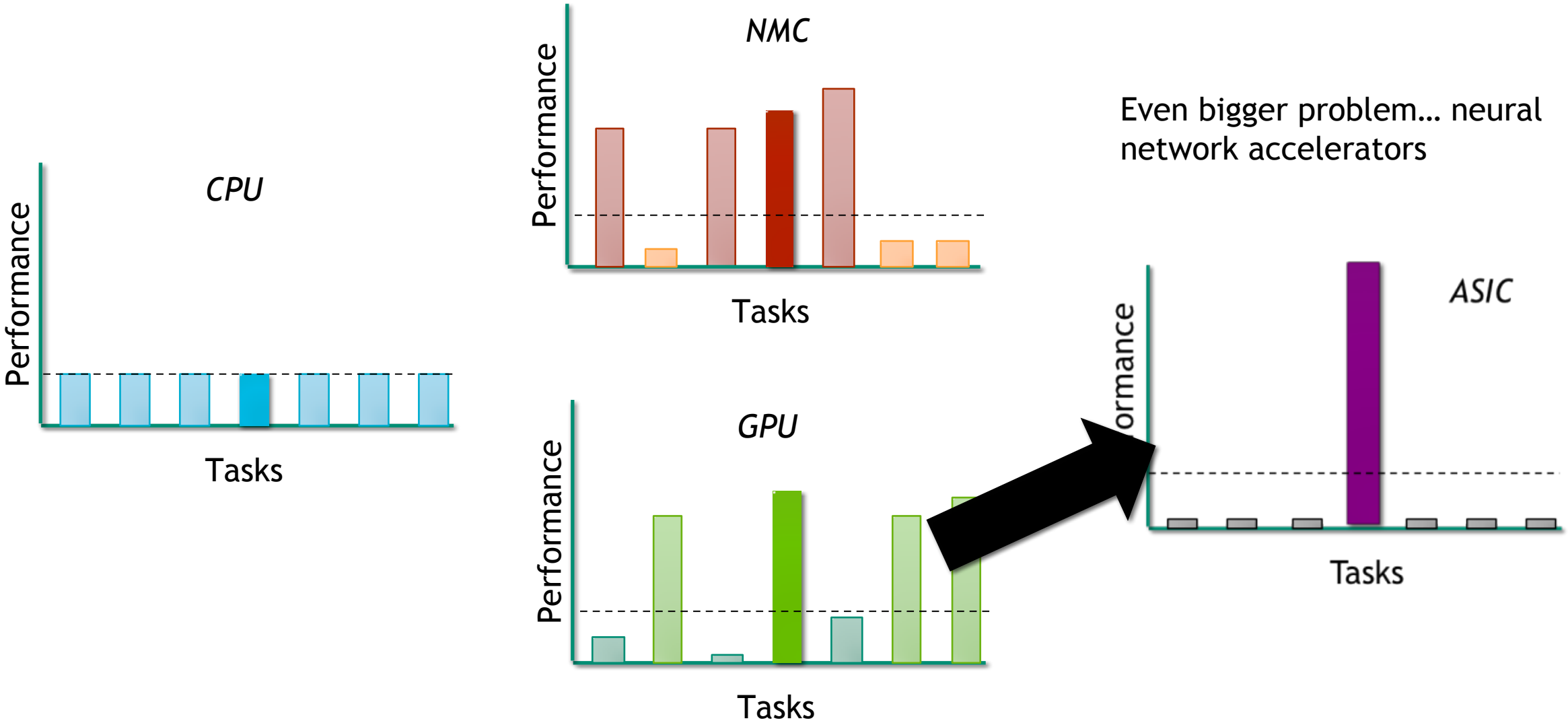






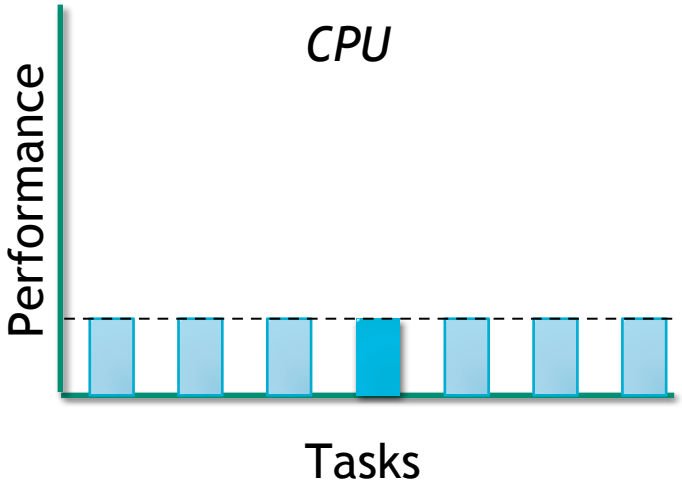
- Yay! Deep learning and neural networks look like a pretty natural fit for NMC!
- BUT… deep learning and neural networks are also a *very* natural fit to GPUs. Perhaps even a better fit…
- We shouldn't be surprised… GPUs are more mature technology and deep learning revolution is largely due to GPUs being good at them

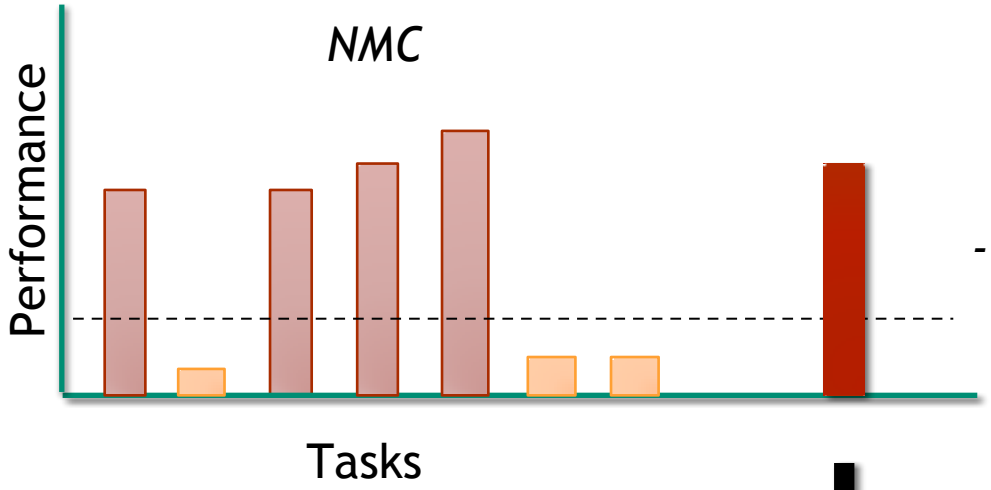
The problem with deep learning and NMC's "killer app"…



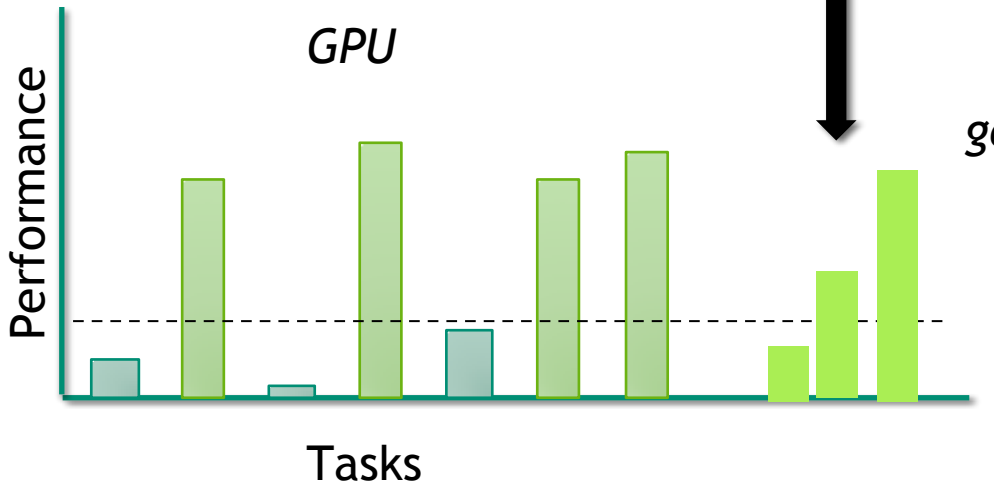
Even bigger problem… neural network accelerators

Option 1: We could look for a novel function well suited for NMC!



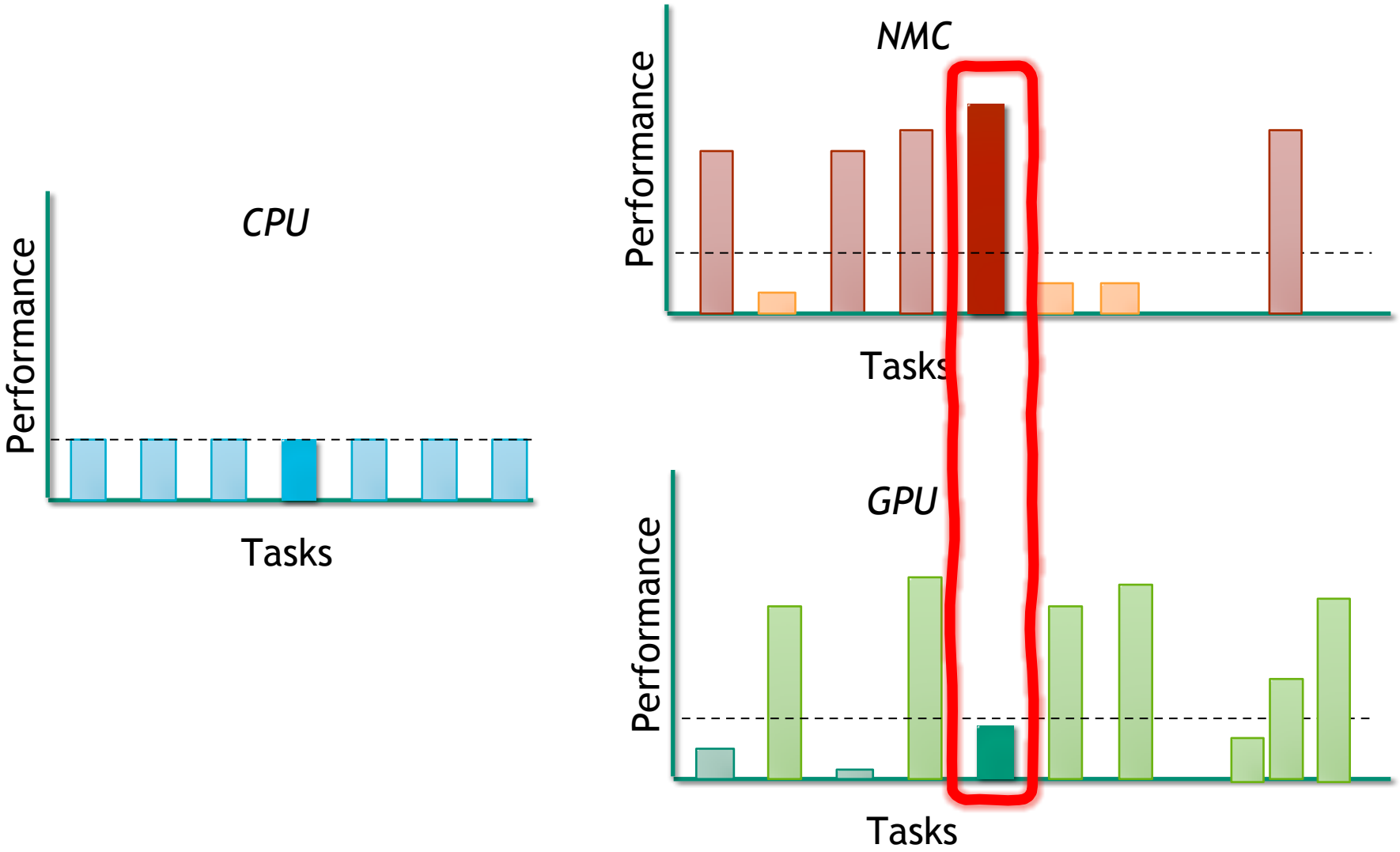


Challenges:
-- Have to create a market and trust in technique



-- Some of these will be good with GPUs, some will not!

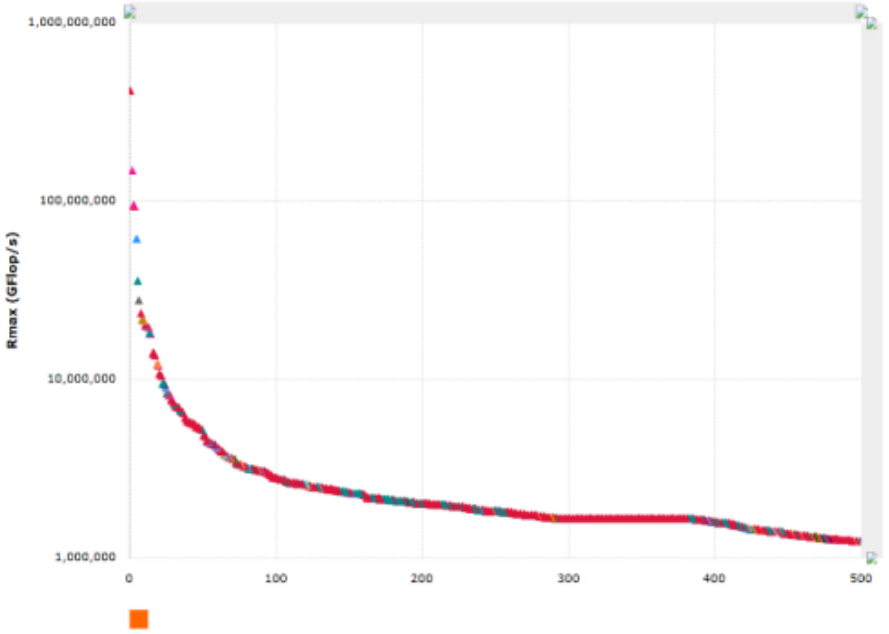
Option 2: We could look at where GPUs do not offer advantage

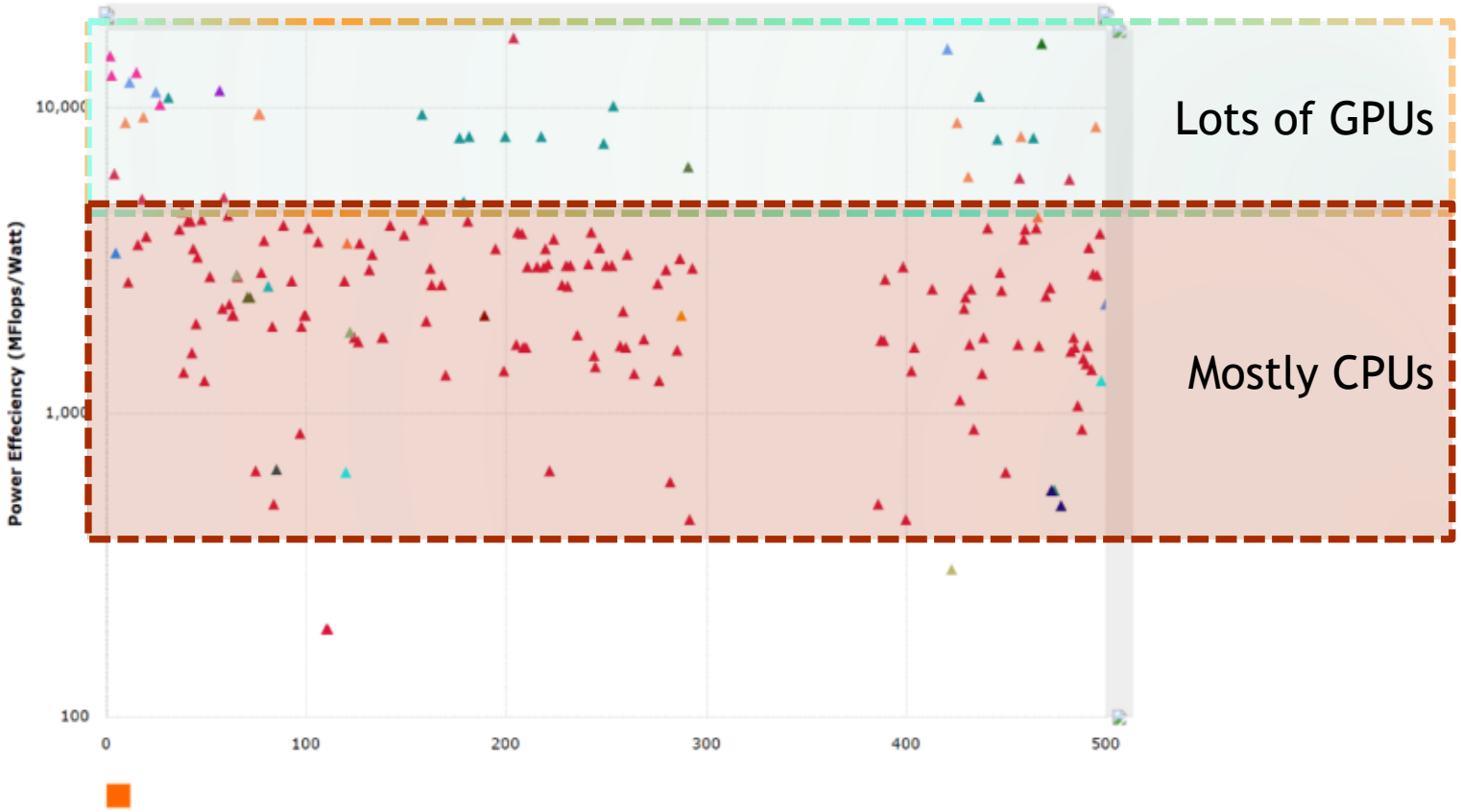


Motivation: Specialization is HPC community's primary path to energy efficiency

Top 500, June 2020

- ❑ Increasing energy-efficiency is critical to reach exascale
- ❑ Biggest systems are leveraging accelerators (mostly GPUs) to increase cores efficiently

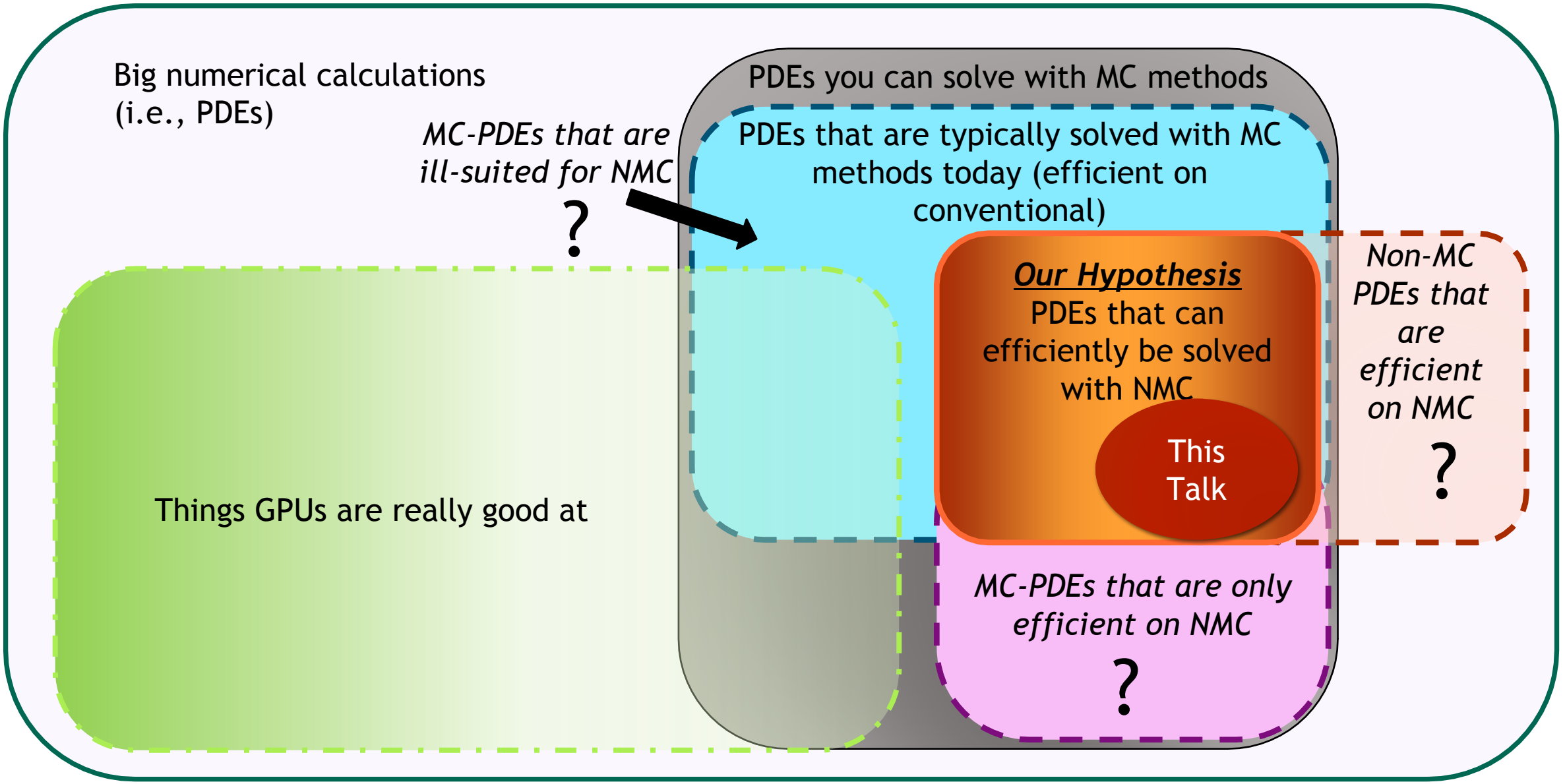




Legend: *Colored by Accelerators / Co-Processors*

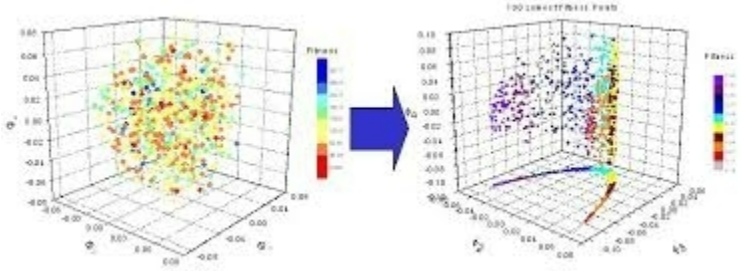
NVIDIA Tesla P100, NVIDIA Tesla V100 SXM2, None, NVIDIA 2050, Intel Xeon Phi 5120D, NVIDIA Tesla V100, Nvidia Volta V100, PEZY-SC2 700Mhz, NVIDIA Tesla K20m, Intel Xeon Phi 7120P, MN-Core, AMD Vega 20, NVIDIA Tesla K40, NVIDIA Quadro RTX8000, Intel Xeon Phi 31S1P, NVIDIA Tesla K40m, Intel Xeon Phi 5110P, NVIDIA Tesla K80, None, NVIDIA Tesla K40/Intel

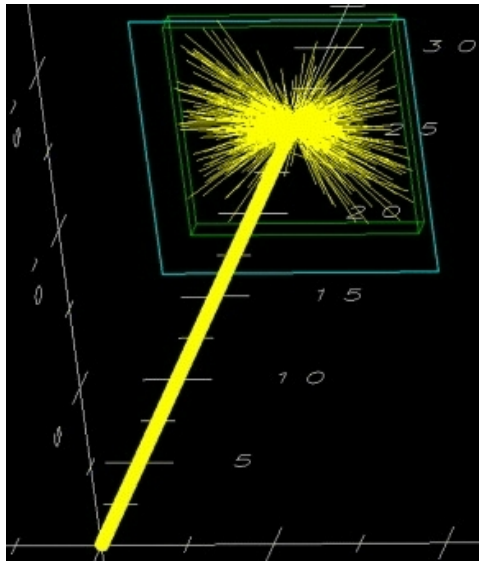
Our hypothesis: There exists a class of scientific computing algorithms for which neuromorphic computing is *efficient*

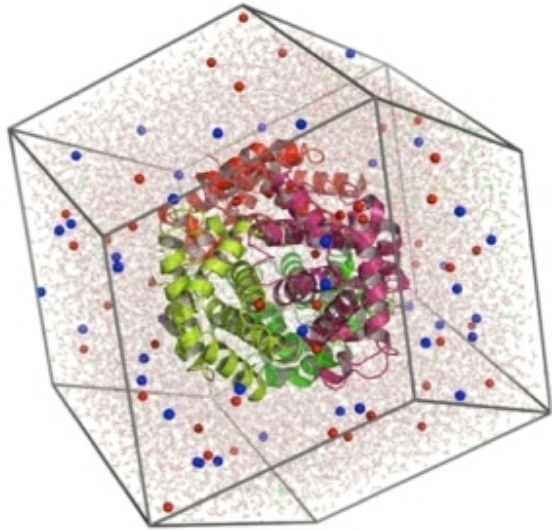


Monte Carlo Approximations of PDE Solutions

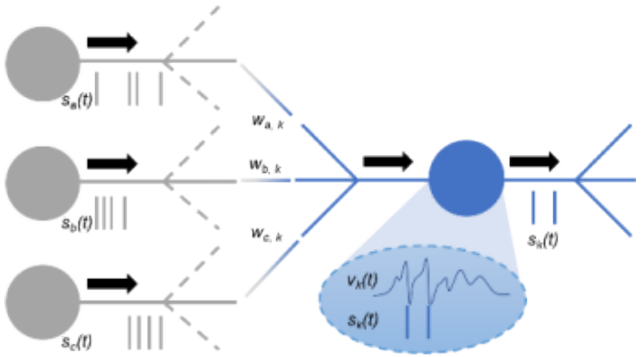
Diffusion: $\frac{\partial C(x,t)}{\partial t} = D\frac{\partial^2 C(x,t)}{\partial x^2}$



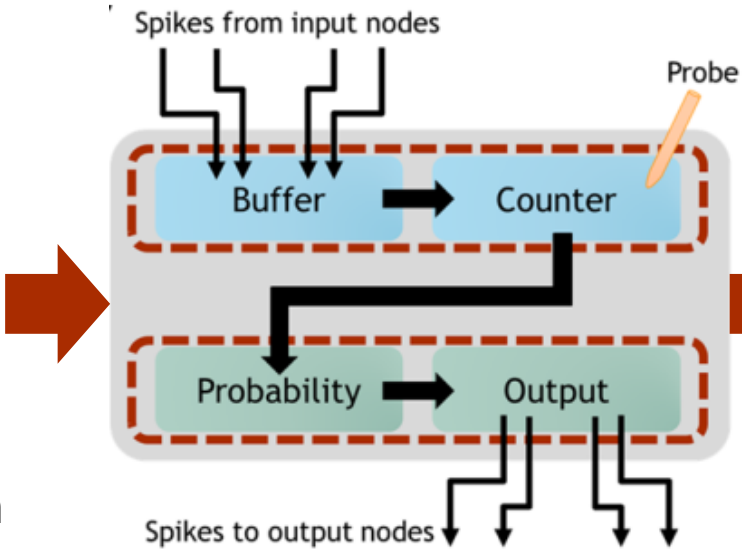


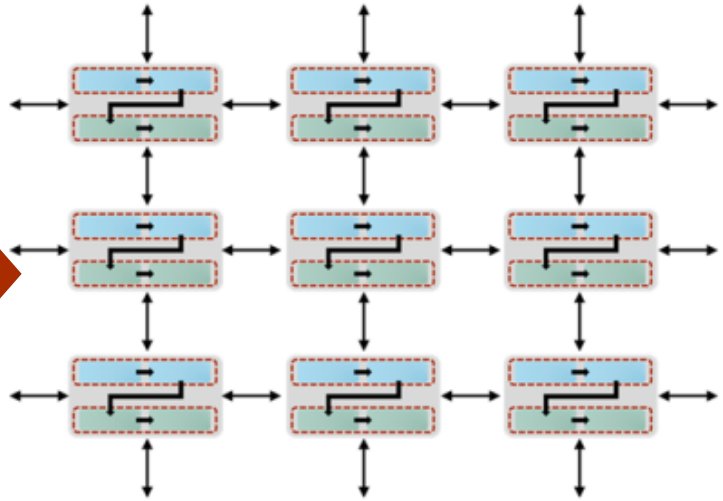


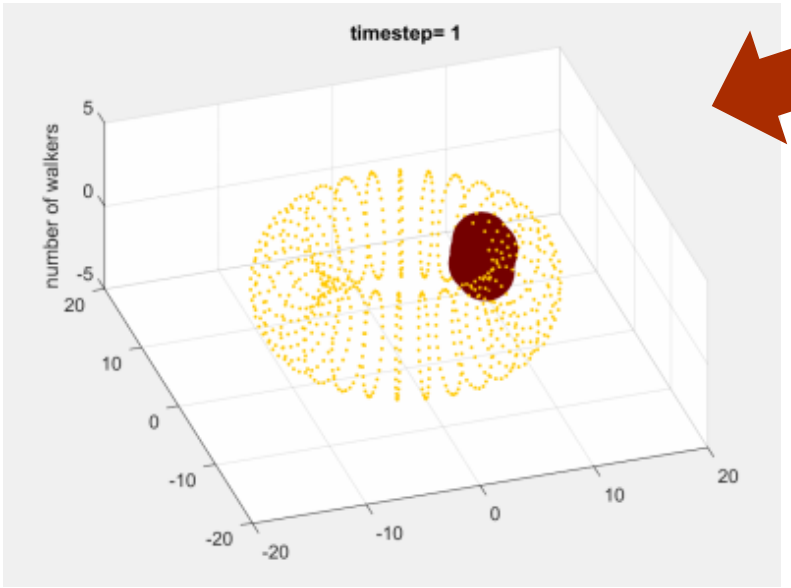
Neuromorphic algorithm can simulate random walks



Leaky Integrate and Fire Neuron



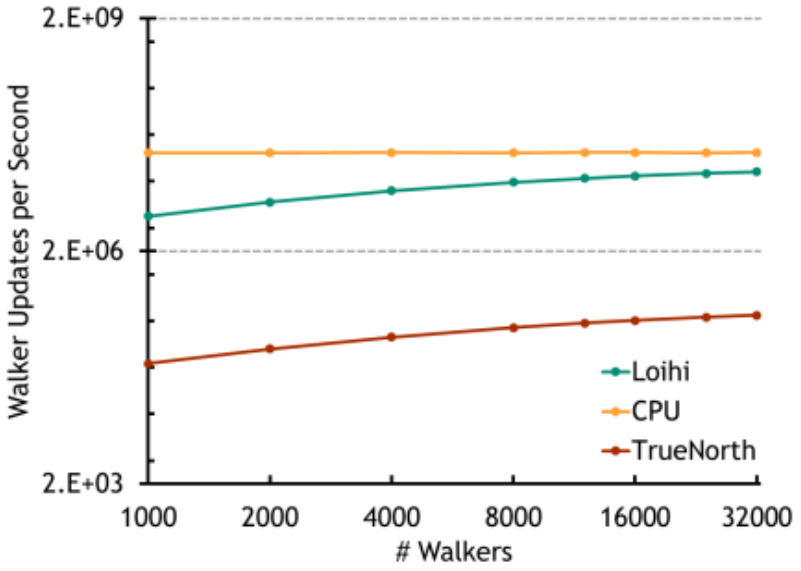




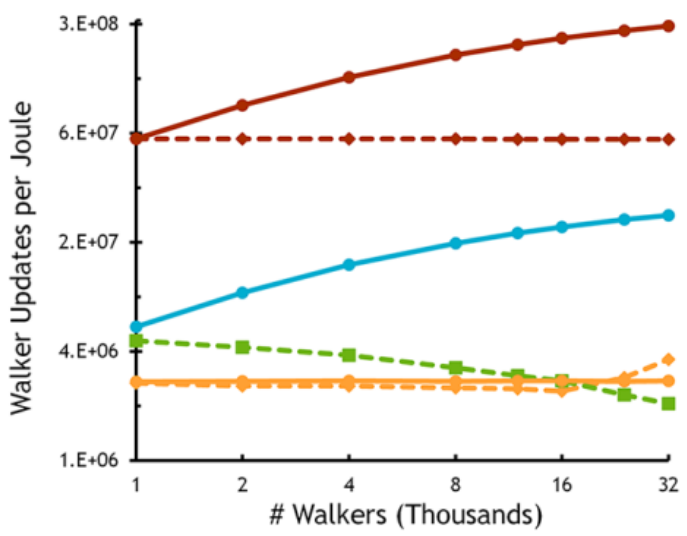
Smith et al., in review 2021

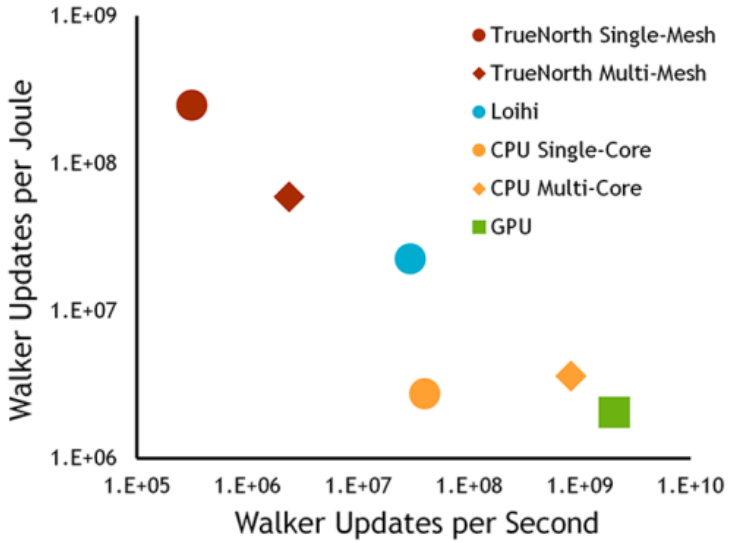
We can identify a neuromorphic advantage for simulating random walks

We define a *neuromorphic advantage* as an algorithm that shows a demonstrable **advantage** in terms of one resource (e.g., energy) while exhibiting comparable **scaling** in other resources (e.g., time).



- We show a *neuromorphic advantage* for implementing simple random walks on neuromorphic hardware compared to CPU implementation
  - Same task, architecture specific algorithms
  - TrueNorth and Loihi are slower, but NMC algorithm time scales better
  - **Overall energy consumption (speed / power) is markedly better (20x-100x) on NMC**





Smith et al., in review 2021

Math: What PDEs can these stochastic processes be useful for?

Class of Partial Integro-Differential Equations:

$$\frac{\partial u}{\partial t}(x,t) = \frac{1}{2} \sum_{i,j} (\sigma\sigma^{T})_{ij}(x,t) \frac{\partial^2}{\partial x_i \partial x_j} u(x,t) + \sum_{i} \mu_i(x,t) \frac{\partial}{\partial x_i} u(x,t)$$

$$+ \lambda(x,t) \int_{\mathbb{R}^d} J(x,y,t) - u(x,t) \, \nu(x,t)(y,t)\,dy$$

$$+ r(x,t) u(x,t) + f(x,t), \qquad x \in \mathbb{R}^d, t \in [0,\infty).$$

Stochastic Process:

NMC Hardware Simulates This Stochastic Process

$$dX(t) = \mu(X(t), t)\, dt + \sigma(X(t), t)\, dB(t) + \int_{\mathbb{R}^d} \gamma(X(t), z)\, dN(t, z)$$

Solution to initial value problem (u(0,x)=g(x)):

Monte Carlo Approximates This Expectation

$$u(x,t) = \mathbb{E}\left[g(X(t)) \exp\left(\int_0^t r(X(s), s)\, ds \right) + \int_0^t f(X(s), s) \exp\left(\int_0^s r(X(\ell), \ell)\, d\ell \right) ds \,\middle|\, X(0) = x \right].$$

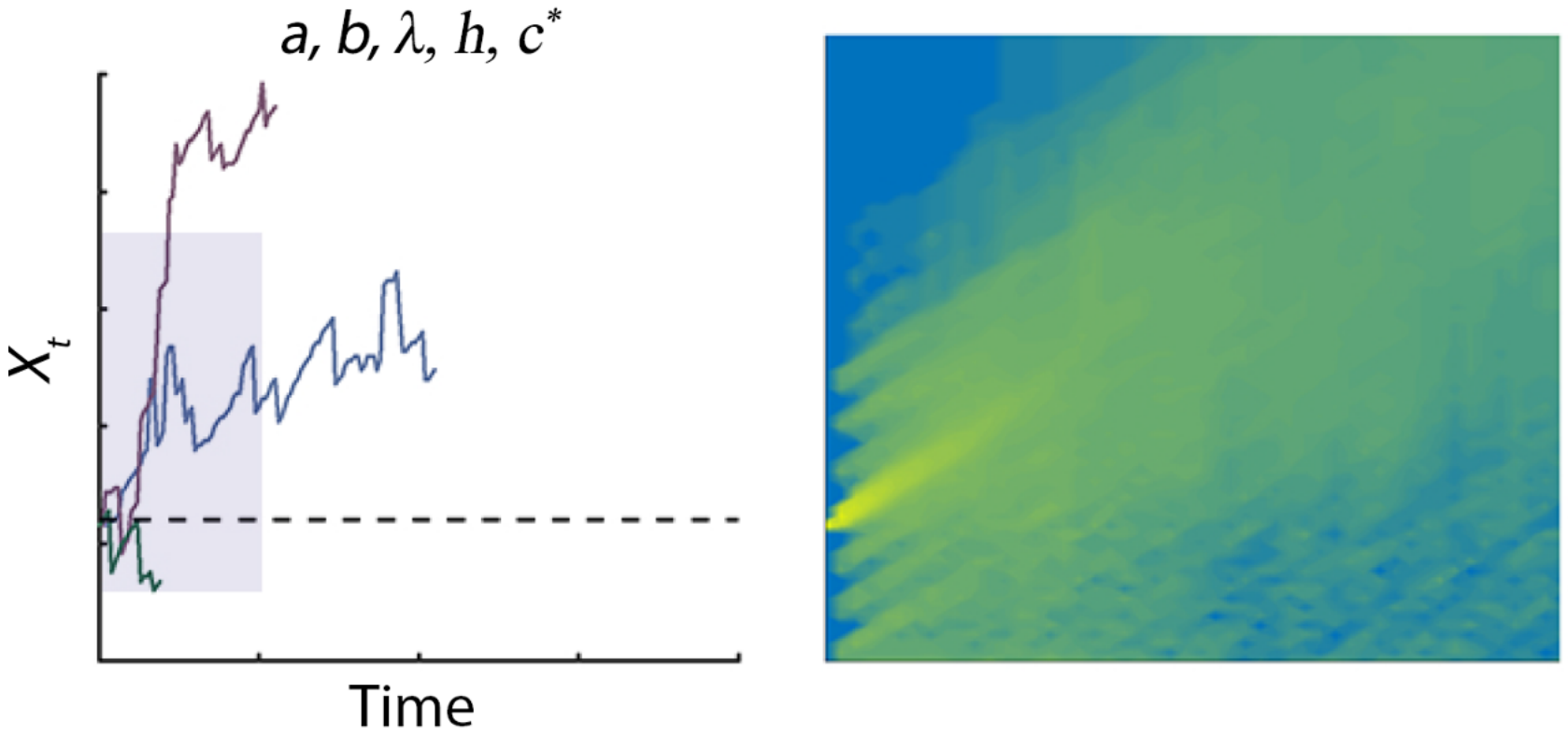
Neural MC algorithm can run wide range of stochastic processes

$$\frac{\partial}{\partial t}\rho(x,t) = \frac{1}{2}\sum_{i,j}^{d}(\sigma\sigma^{T})_{ij}(x,t)\frac{\partial^2}{\partial x_i \partial x_j}\rho(x,t) + \sum_{i}^{d}\mu_i(x,t)\frac{\partial}{\partial x_i}\rho(x,t)$$

$$-\rho(x,t)\int_{\mathbb{R}^d} r(x,y,t)\,dy + \int_{\mathbb{R}^d} r(x,y,t)\,\rho(y,t)\,dy$$

$$-\rho(x,t)\int_{\mathbb{R}^d}\rho(y,t)\,dy + \rho(x,t), \qquad x\in\mathbb{R}^d,\ t\in[0,\infty).$$

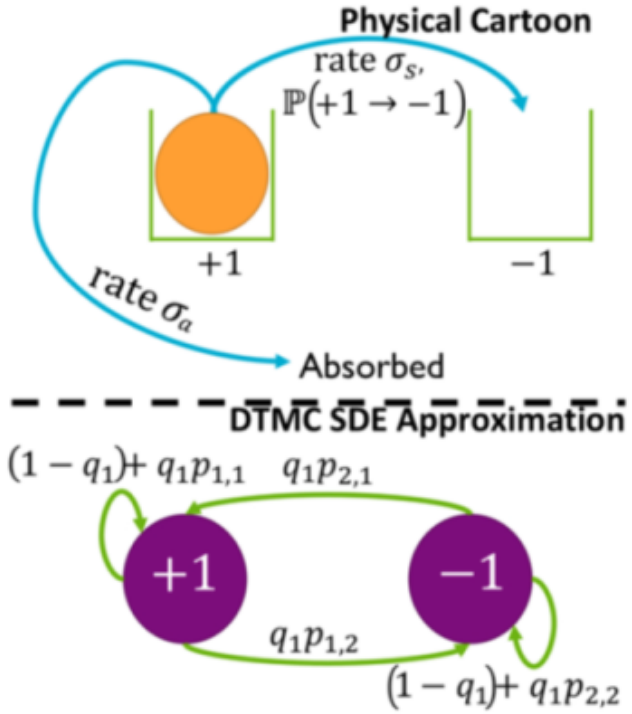
$$dX(t) = \mu(X(t),t)\,dt + \sigma(X(t),t)\,dW(t) + \int_{\mathbb{R}^d} c(X(t),z)\,dN(X(t),z,t)\,.$$

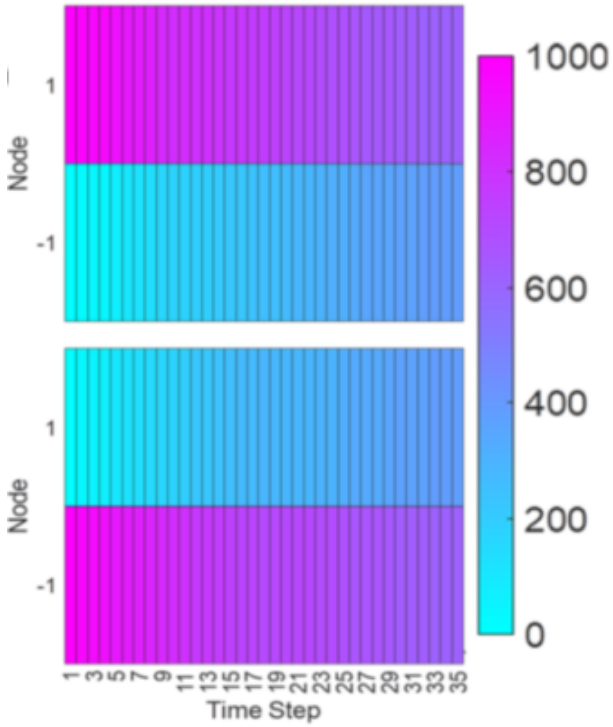


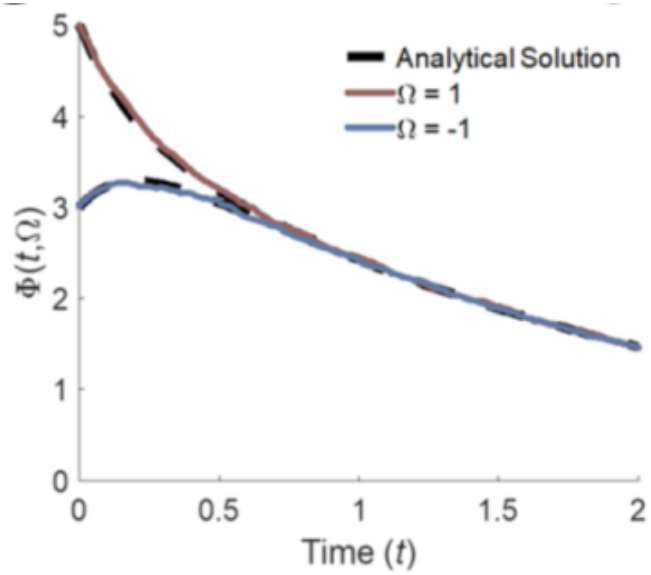
Smith et al., in review 2021

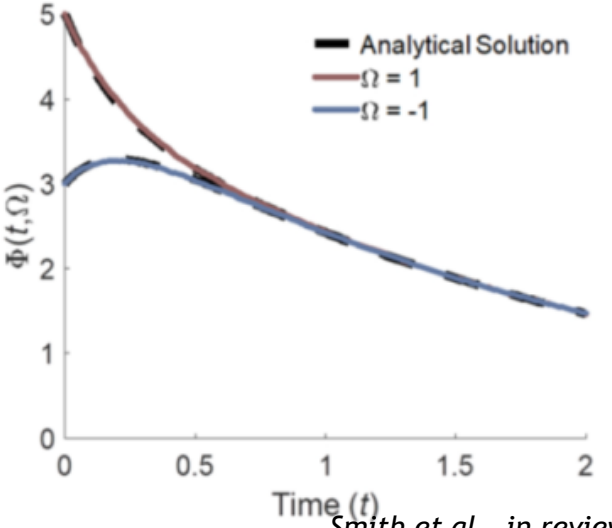
Some more applied examples

- Boltzmann state transition
  - Particle can exist in 2 states (+1 or -1) or be absorbed.
  - Implement as simple stochastic process on TrueNorth





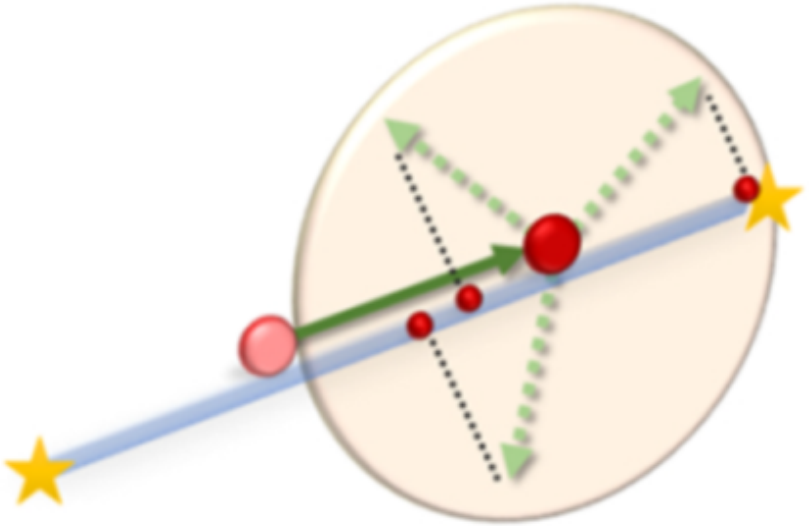


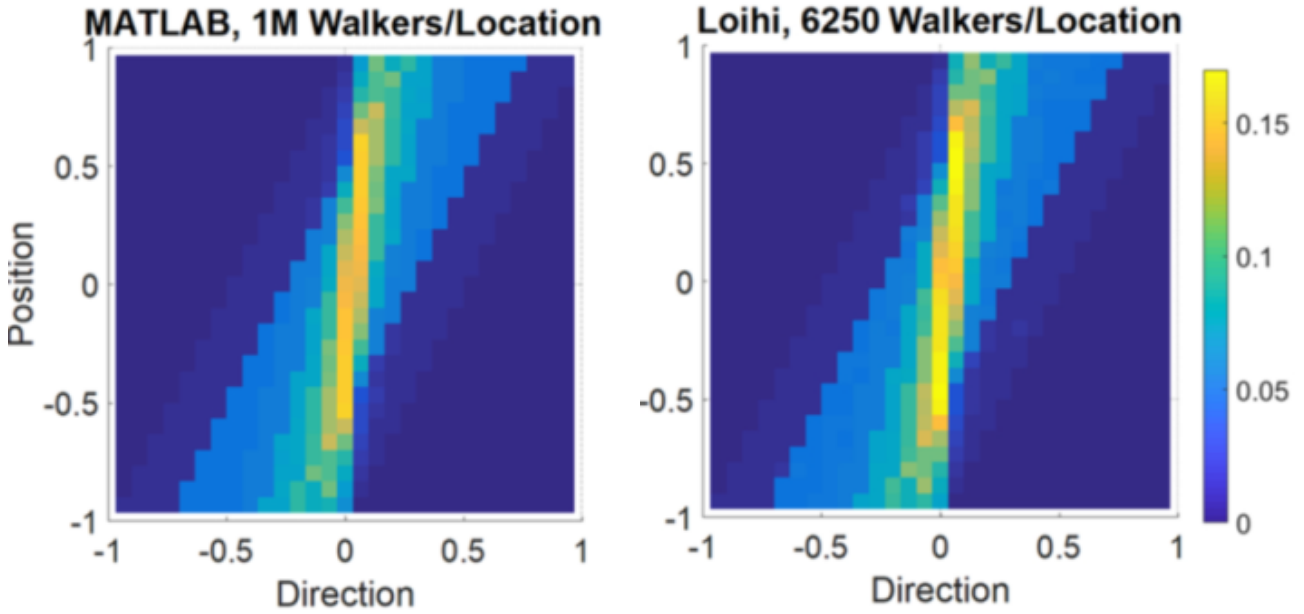


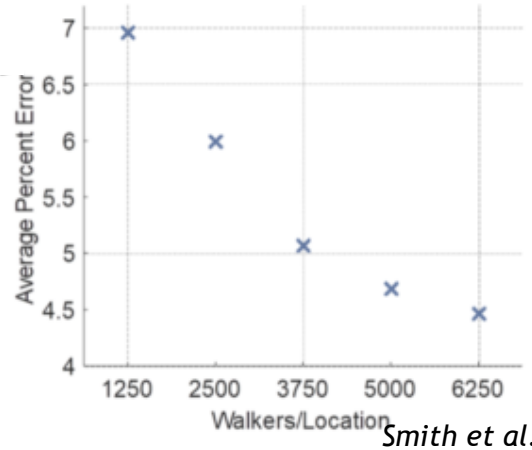
Smith et al., in review 2021

Some more applied examples

- 1D particle transport
  - Particle moves in 2D, only track 1D.
  - At point x=0, particle reflects in random direction
  - Track velocity in x-dimension and angle
  - Implemented on Loihi



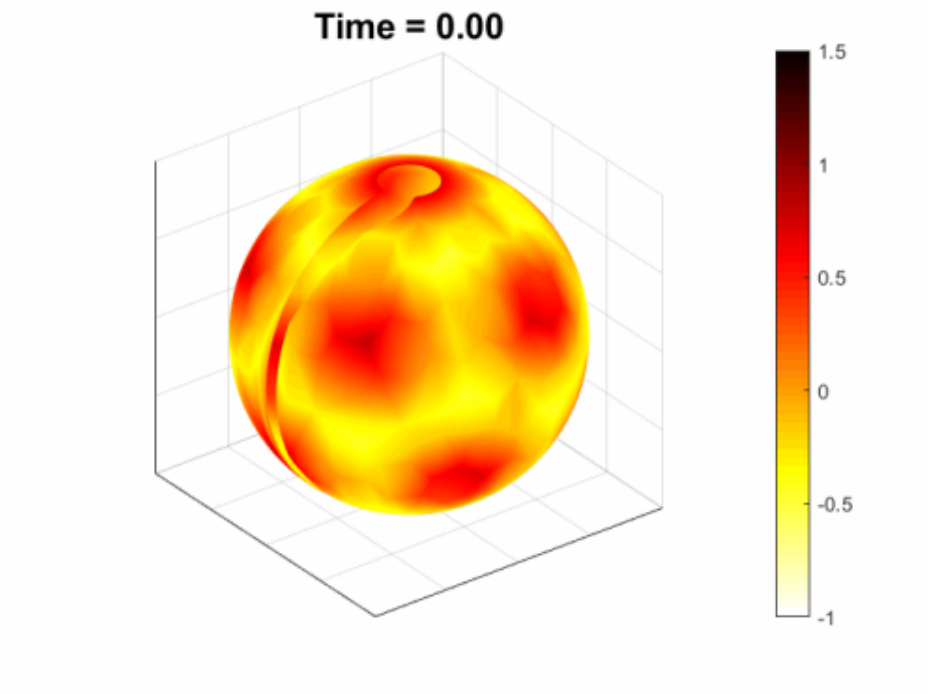


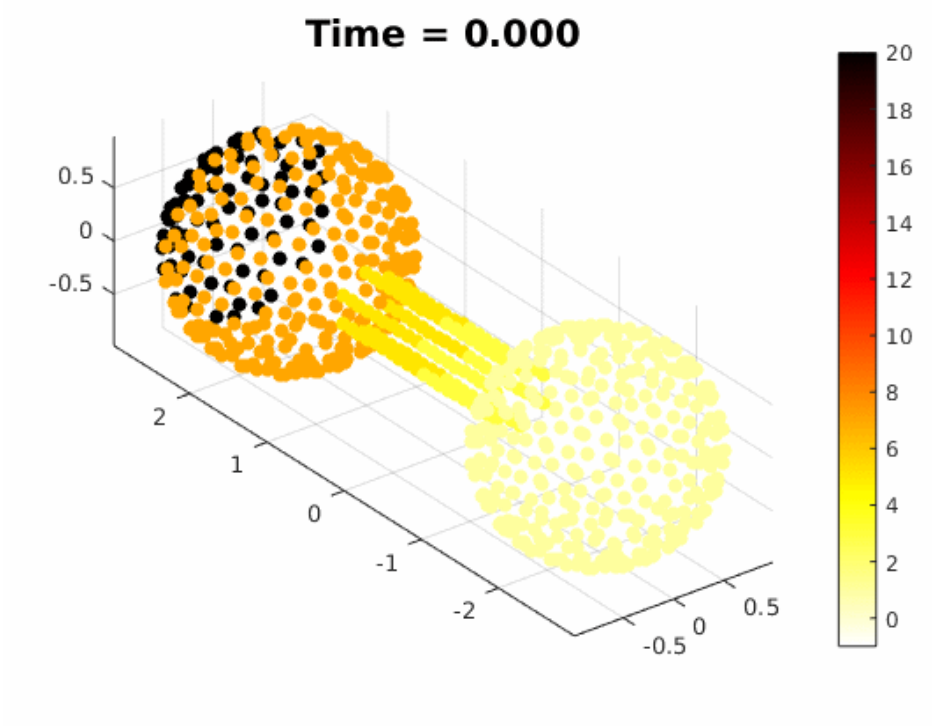


Smith et al., in review 2021

Some more applied examples

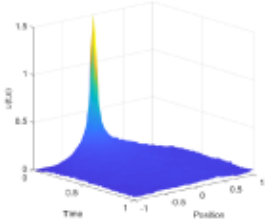
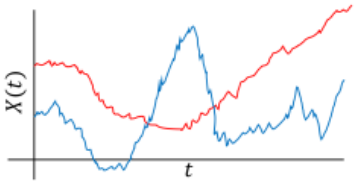
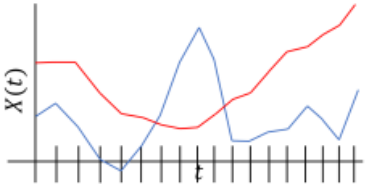
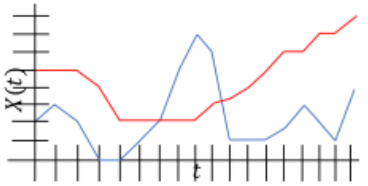
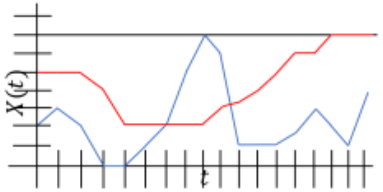
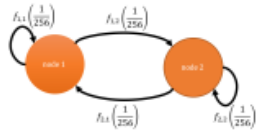
- Heat transport on surface of sphere and barbell
  - Not a trivial diffusion surface, complex stochastic process and transition probabilities
  - Implemented sphere on Loihi; barbell in simulation





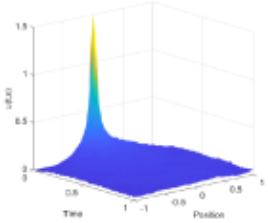
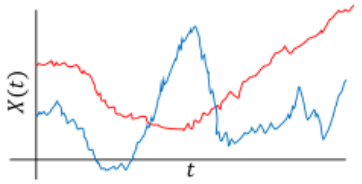
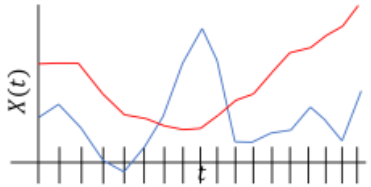
Smith et al., in review 2021

Accuracy Stack for Neuromorphic Implementation

PDE Ground Truth	$u_t = f(t\mu\mu_x\mu_{xx})$ $u(t,x) = \mathbb{E}[g(t,X_t)\|X_0 = x]$		
Problem	**Approximation**	**Visualization**	**Error/Convergence**
To approximate the expectation, we must sample paths of the stochastic process.	$u(t,x) \approx \frac{1}{M}\sum_{i=1}^{M} g(t, X_t^i);\ X_0^{(i)} = x$		$\frac{1}{\sqrt{M}}$
Continuous paths cannot be sampled, we must employ a discretization scheme.	$u(j\Delta t,x) \approx \frac{1}{M}\sum_{i=1}^{M} g(j\Delta t, X_{j\Delta t}^i);\ X_0^{(i)} = x$		$\sqrt{\Delta t}$
Neurons cannot represent a continuum of locations. Hence we must limit the spatial locations of the walk.	$u(j\Delta t,x_k) \approx \frac{1}{M}\sum_{i=1}^{M} g(j\Delta t, \hat{X}_{j\Delta t}^i);\ \hat{X}_0^{(i)} = x_k$		$\frac{1}{2}j\Delta t\Delta s$
There are a finite number of neurons, so maximum and minimum values for the random walk will exist.	$u(j\Delta t,x_k) \approx \frac{1}{M}\sum_{i=1}^{M} g(j\Delta t, \check{X}_{j\Delta t}^i);\ \check{X}_0^{(i)} = x_k$		varies
Hardware Specific Issues. TrueNorth having quantized probability, for example.	$\mathbb{P} \propto \frac{1}{256}$		varies

Smith et al., in review 2021

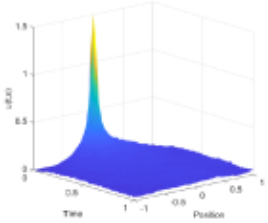
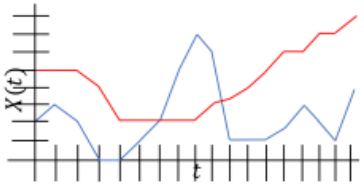
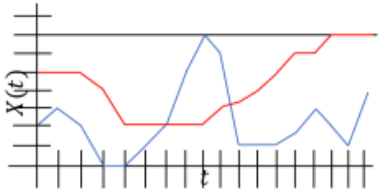
Accuracy Stack for Neuromorphic Implementation

PDE Ground Truth	$u_t = f(t\mu\mu_x\mu_{xx})$ $u(t,x) = \mathbb{E}[g(t,X_t)\|X_0 = x]$		
Problem	**Approximation**	**Visualization**	**Error/Convergence**
To approximate the expectation, we must sample paths of the stochastic process.	$u(t,x) \approx \frac{1}{M}\sum_{i=1}^{M} g(t, X_t^i);\ X_0^{(i)} = x$		$\frac{1}{\sqrt{M}}$
Continuous paths cannot be sampled, we must employ a discretization scheme.	$u(j\Delta t,x) \approx \frac{1}{M}\sum_{i=1}^{M} g(j\Delta t, X_{j\Delta t}^i);\ X_0^{(i)} = x$		$\sqrt{\Delta t}$

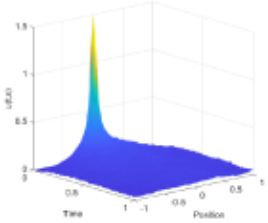
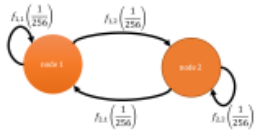
⇧ ⇧ ⇧ ⇧ ⇧ ⇧

Present for any implementation

Accuracy Stack for Neuromorphic Implementation

Problem	Approximation	Visualization	Error/Convergence
PDE Ground Truth	$u_t = f(t\mu\mu_x\mu_{xx})$ $u(t,x) = \mathbb{E}[g(t,X_t)\|X_0 = x]$		
Neuromorphic Specific ⇩ ⇩ ⇩ ⇩ ⇩ ⇩			
Neurons cannot represent a continuum of locations. Hence we must limit the spatial locations of the walk.	$u(j\Delta t, x_k) \approx \frac{1}{M}\sum_{i=1}^{M} g(j\Delta t, \hat{X}^i_{j\Delta t});\ \hat{X}^{(i)}_0 = x_k$		$\frac{1}{2}j\Delta t\Delta s$
There are a finite number of neurons, so maximum and minimum values for the random walk will exist.	$u(j\Delta t, x_k) \approx \frac{1}{M}\sum_{i=1}^{M} g(j\Delta t, \check{X}^i_{j\Delta t});\ \check{X}^{(i)}_0 = x_k$		varies
⇧ ⇧ ⇧ ⇧ ⇧ ⇧			

Accuracy Stack for Neuromorphic Implementation

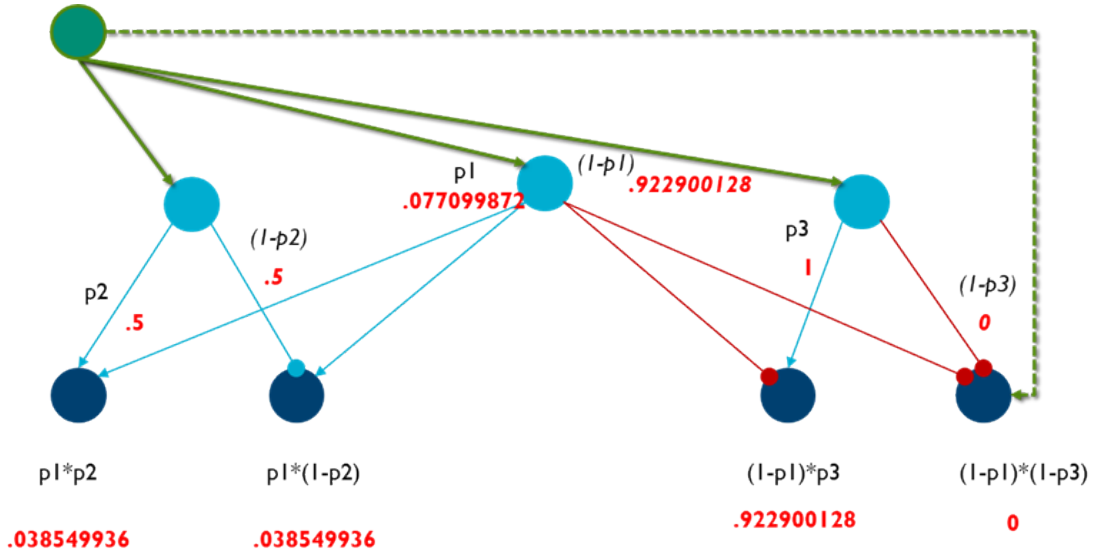
Problem	Approximation	Visualization	Error/Convergence
PDE Ground Truth	$u_t = f(t\mu\mu_x\mu_{xx})$ $u(t,x) = \mathbb{E}[g(t,X_t)\|X_0 = x]$		
Hardware Specific Issues. TrueNorth having quantized probability, for example.	$\mathbb{P} \propto \frac{1}{256}$		varies *Smith et al., in review 2021*

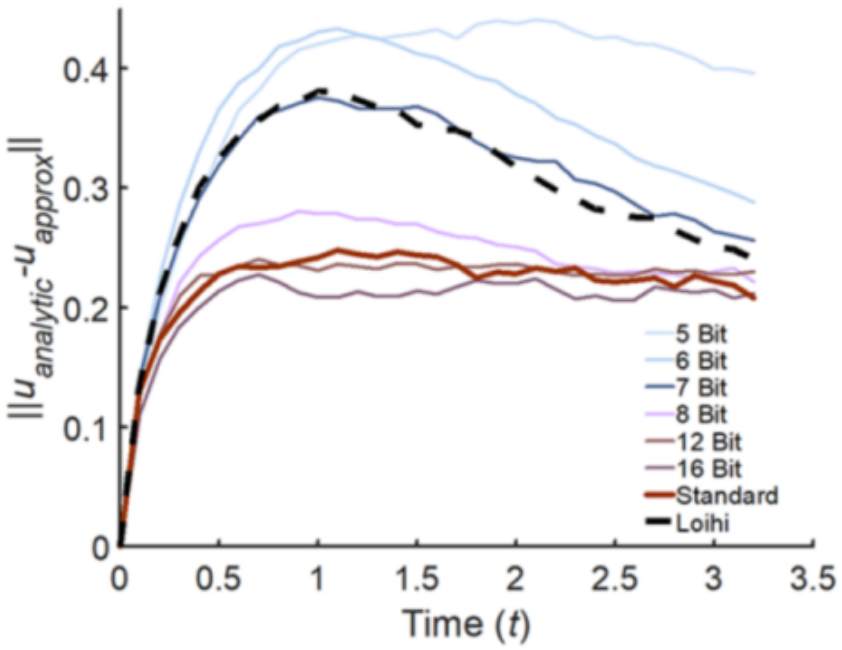
Hardware Specific

⇩ ⇩ ⇩ ⇩ ⇩ ⇩

Precision example: spherical diffusion

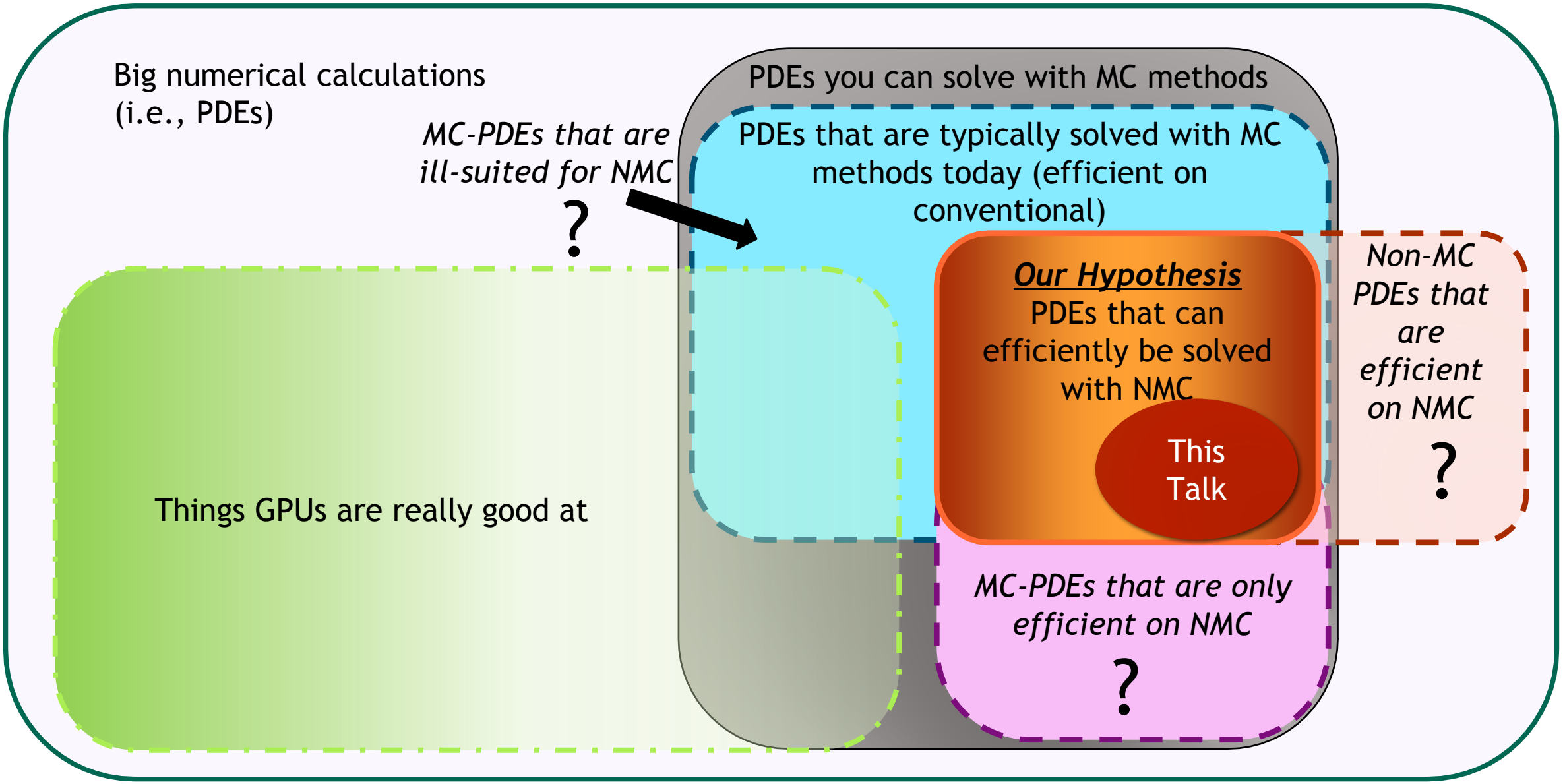
- Transition probabilities between neural mesh points are determined by RNGs on probabilistic neurons
- On Loihi, PRNGs are 8-bit, effectively making transition probabilities 8-bit
- Comparing sphere Loihi example to MATLAB simulation with reduced precision suggests Loihi is roughly 7-bit precision



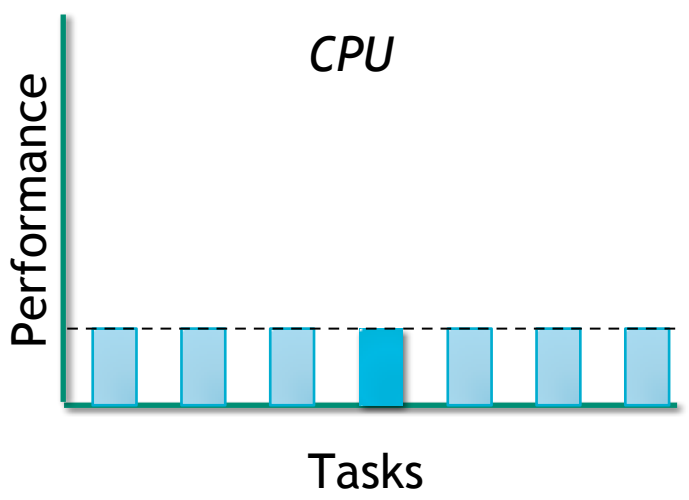


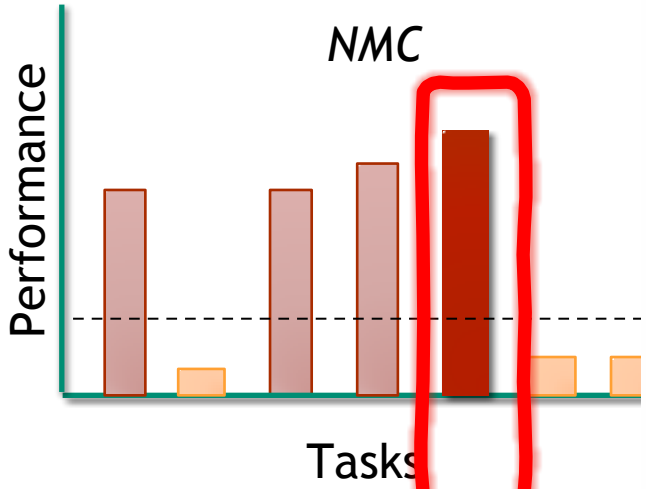
Smith et al., in review 2021

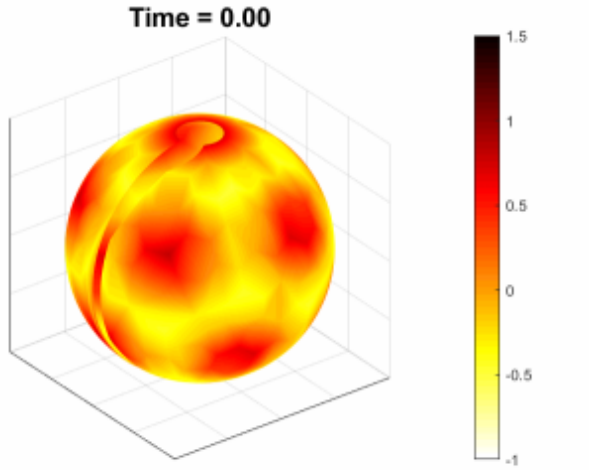
Our hypothesis: There exists a class of scientific computing algorithms for which neuromorphic computing is *efficient*

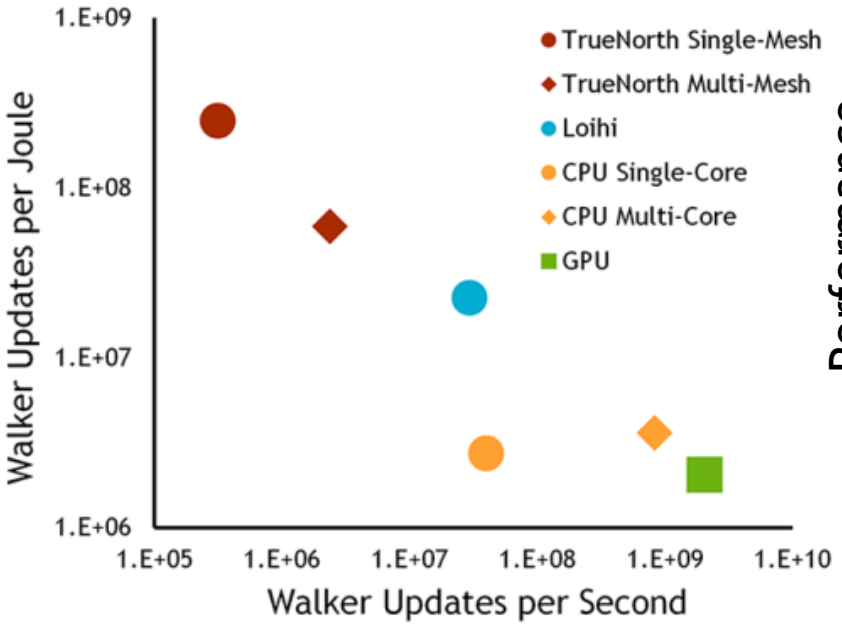


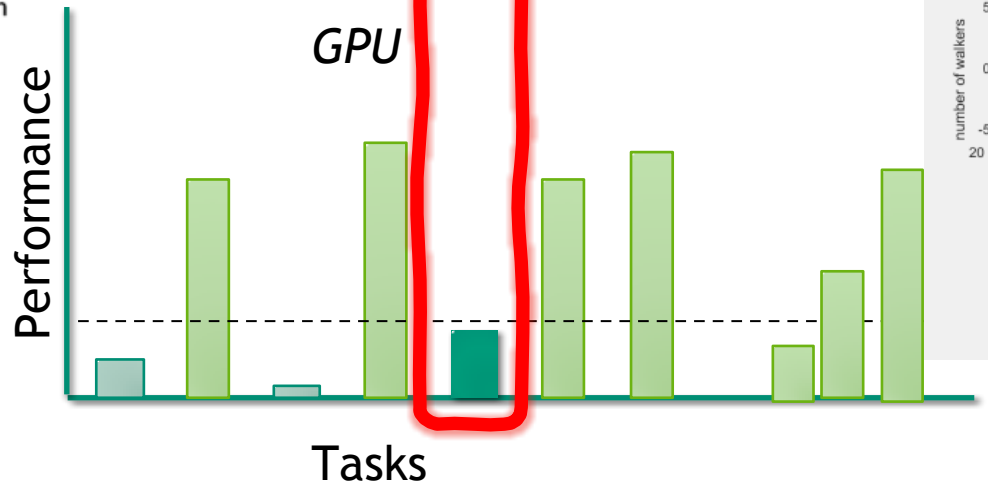
Monte Carlo – A neuromorphic advantage over CPUs and GPUs!

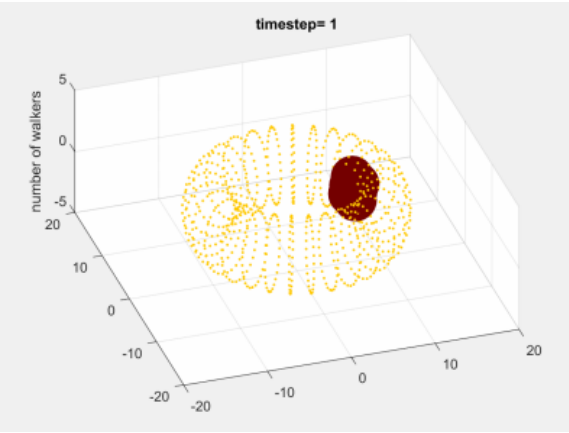












Smith et al., in review 2021

Thanks!

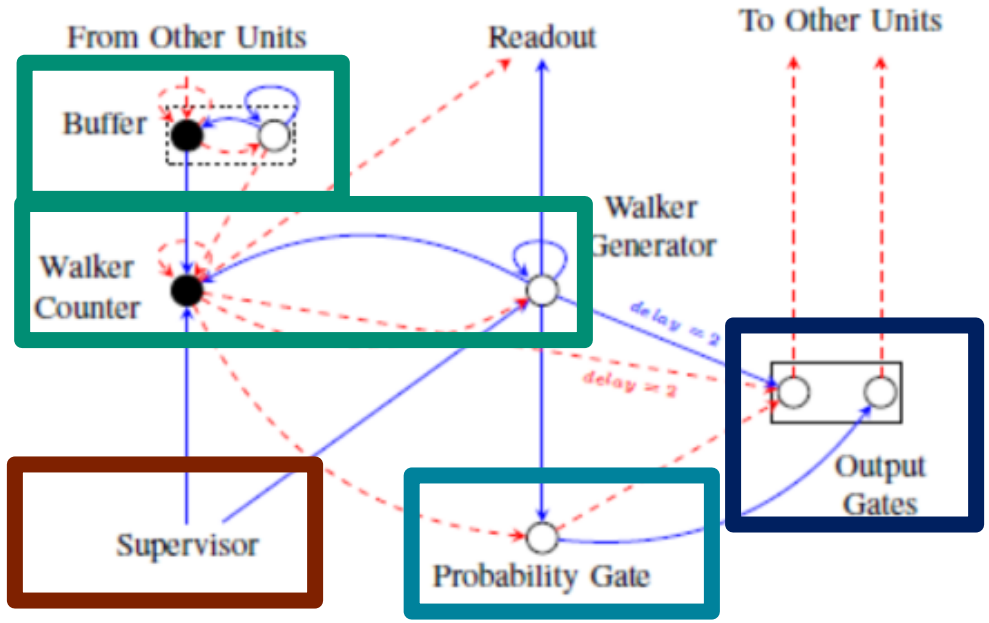
- Sandia Neural PDE team:

  Darby Smith, Aaron Hill, Leah Reeder, Ojas Parekh, William Severa, Brian Franke, Rich Lehoucq

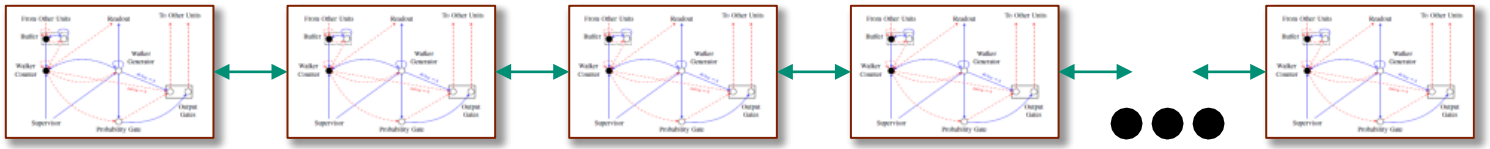
- Neuromorphic Hardware:
  - Loihi Access: Craig Vineyard (SNL), Suma Cardwell (SNL), Intel INRC
  - TrueNorth Access: Lawrence Livermore National Laboratory

- Funding
  - Sandia Laboratory Directed Research and Development
  - DOE Advanced Simulation and Computing

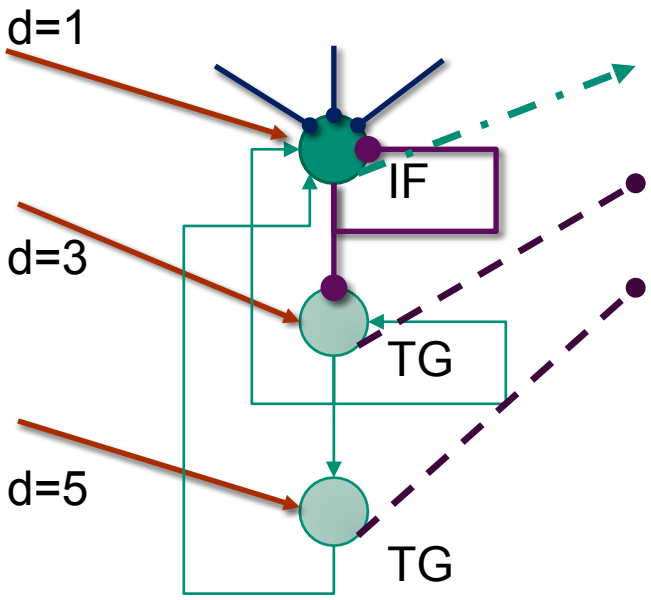
Loihi-specific circuit



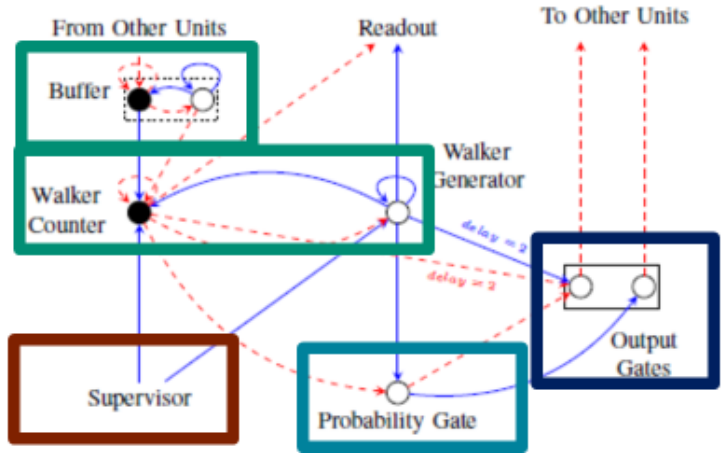
1. Supervisor circuit
   1. Start buffer
   2. Start counter
2. Counter circuit
   1. Buffer neurons
   2. Counter neurons
3. Probabilistic neurons
4. Output neurons

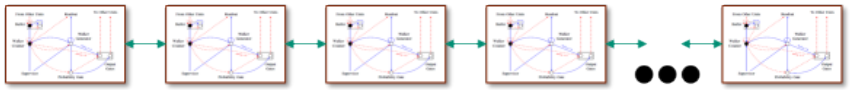


Loihi-specific circuit – node buffer and node counter



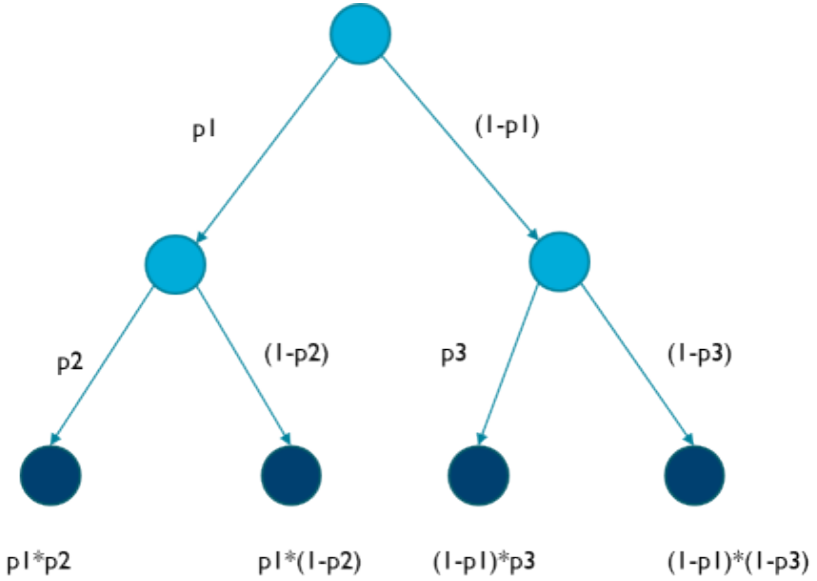
To next counter circuit or output nodes

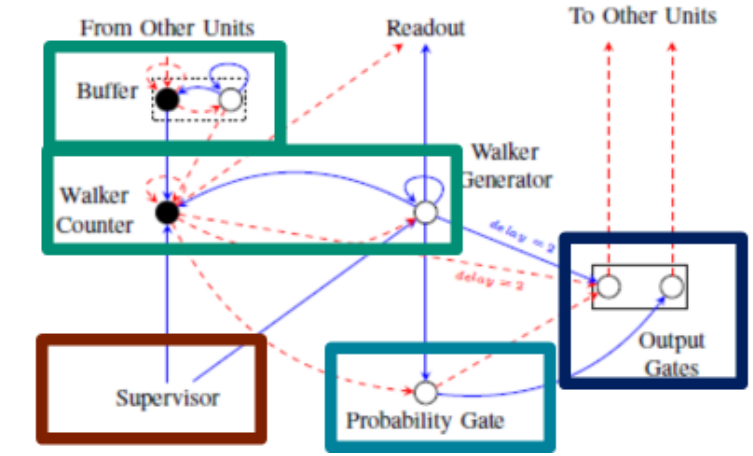


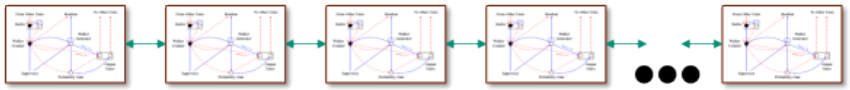


1- COUNTER neuron stores number of walkers as *negative voltage*
2- Supervisor input causes GENERATOR to fire as long as $V_{COUNTER}$ is negative
3- RELAY neuron ensures that $V_{COUNTER}$ is appropriately reset if it pre-emptively shuts off

Loihi-specific circuit – random probabilities and outputs

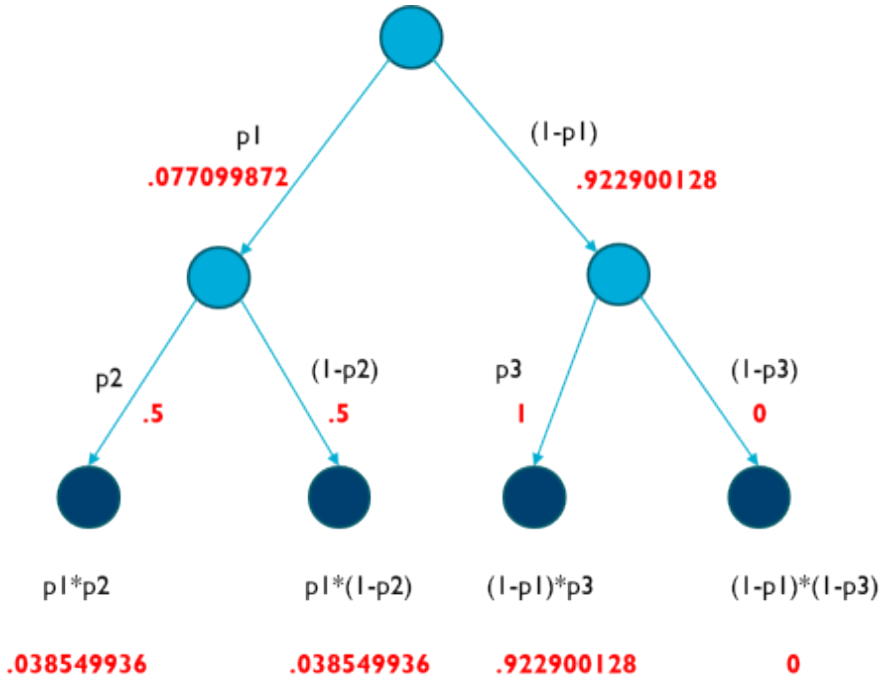


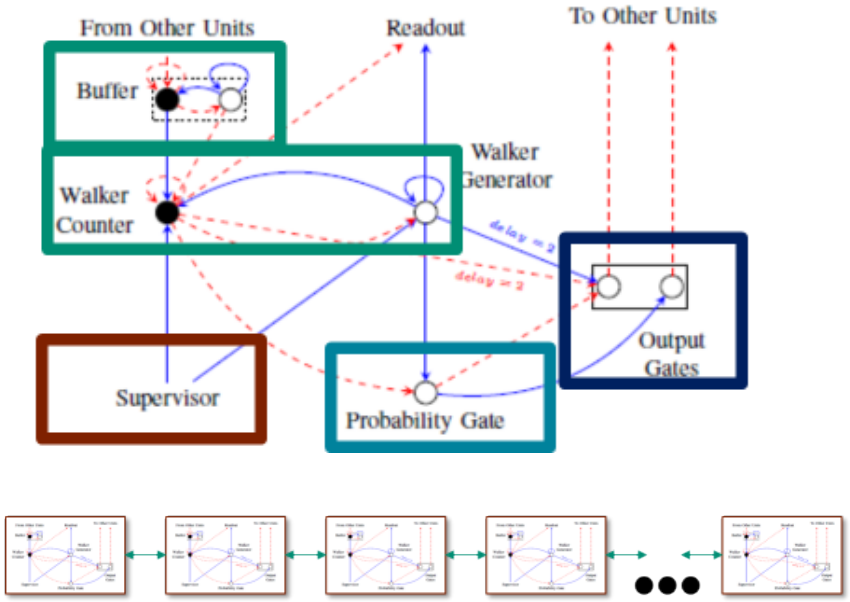




1- Goal: we need to get pre-defined probabilities that the walker gets directed to along the kth output direction
2 – In principle, a tree structure would work, but it does not scale well

Loihi-specific circuit – random probabilities and outputs



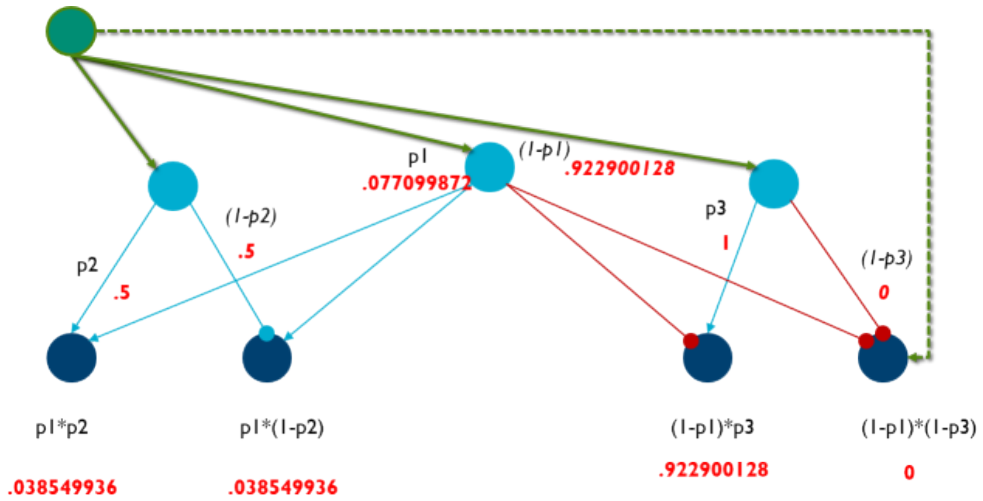


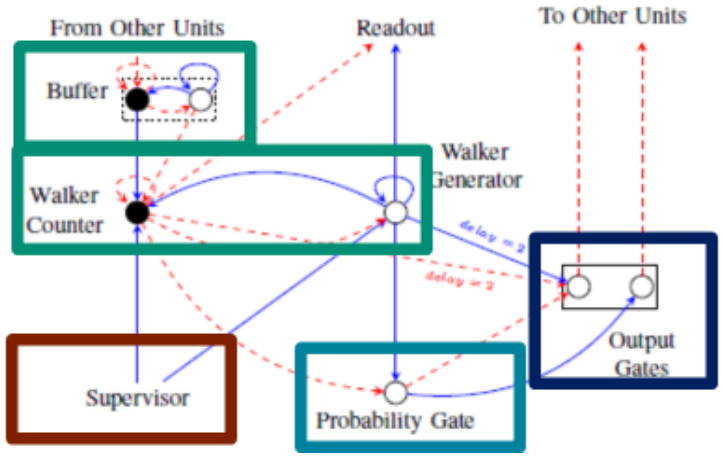
1- Goal: we need to get pre-defined probabilities that the walker gets directed to along the kth output direction

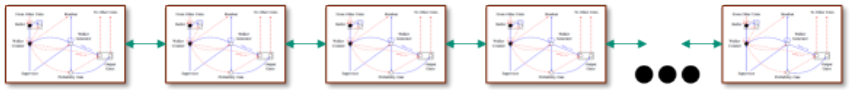
2 – In principle, a tree structure would work, but it does not scale well

- Can use tree to compute probabilities

Loihi-specific circuit – random probabilities and outputs







1- Goal: we need to get pre-defined probabilities that the walker gets directed to along the kth output direction

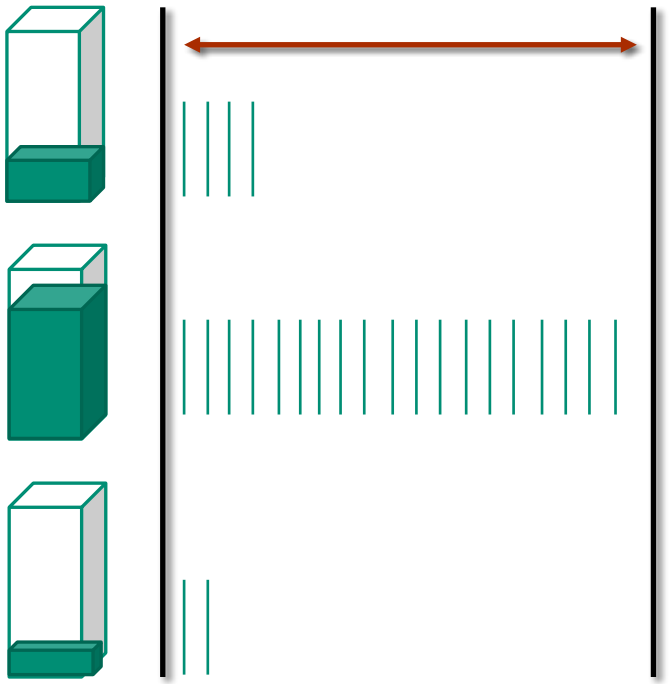
2 – In principle, a tree structure would work, but it does not scale well

- Can use tree to compute probabilities

3 – Can collapse tree into single layer

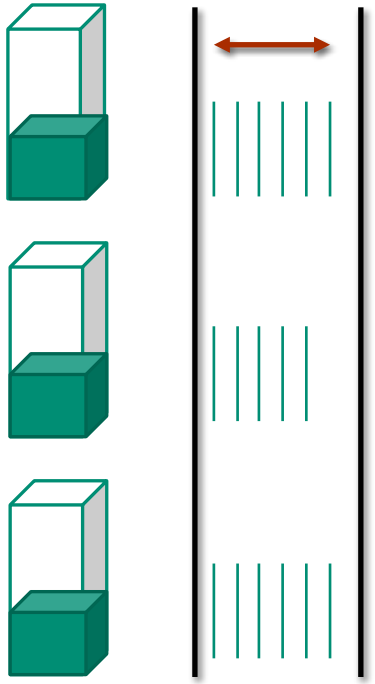
- Single layer tree is both faster and has predictable delays…

Unusual relationship between neural timesteps and simulation timesteps



Uneven distribution of walkers

- More neural timesteps required for simulation update
- More common early in diffusion



Even distribution of walkers

- Fewer neural timesteps required for simulation update
- More common late in diffusion