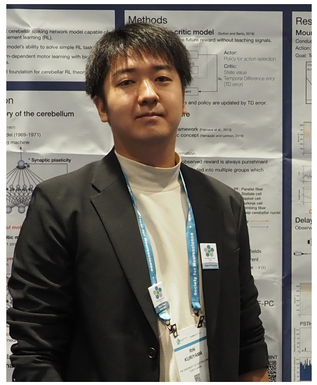
「富岳」を用いた細胞形状を考慮する神経回路の大規模シミュレーション

電気通信大学大学院情報理工学研究科博士三年 山﨑匡研究室
栗山凜

2024年度 第2回 HPCIC計算科学フォーラム 2025年3月31日

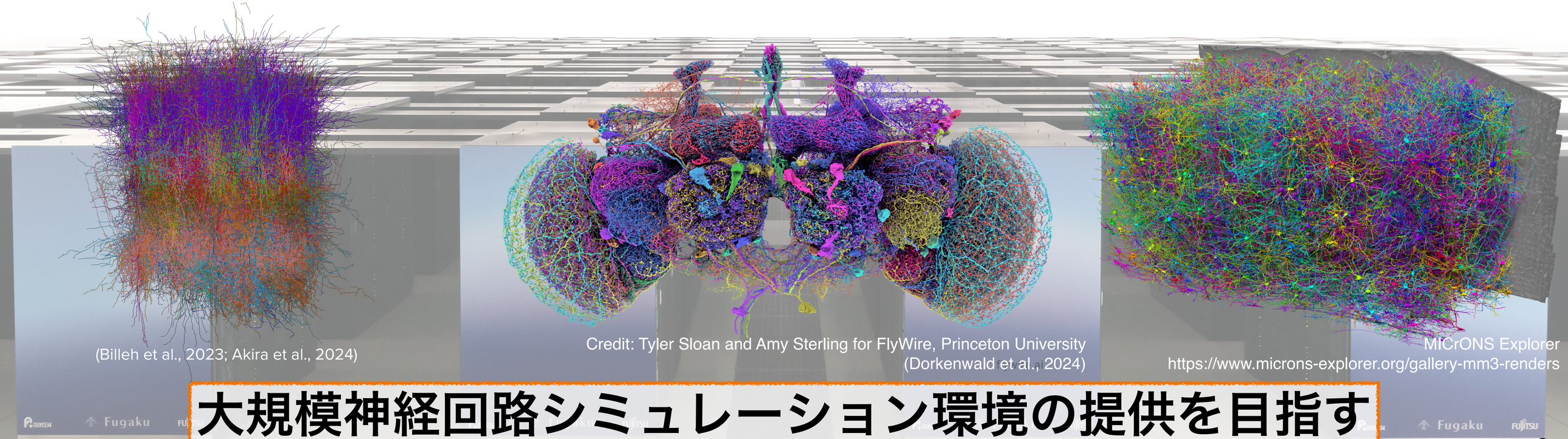
自己紹介



- 電気通信大学大学院情報理工学研究科博士3年
  2019年から山崎匡研究室所属
- 情報工学、神経回路シミュレーション、HPC
- 主な研究内容
  - GPUを用いた小脳神経回路シミュレーションの高速化
    Kuriyama et al., 2021. Kobayashi, Kuriyama, Yamazaki, 2022.
  - 分散計算機上脳身体シミュレーション
    Kuniyoshi et al. 2023.
  - **スーパコンピュータ「富岳」における Neulite シミュレータの並列実装**
    Kuriyama, Akira, Yamazaki. (2024) JNNS2024

概要
大規模神経回路シミュレーション

- ミクロ, メゾスコピックレベルの脳マップ (コネクトーム) が徐々に解明
- 実験では様々な制約：**操作可能な脳モデル** → 脳の情報処理の理解



(Billeh et al., 2023; Akira et al., 2024)

Credit: Tyler Sloan and Amy Sterling for FlyWire, Princeton University
(Dorkenwald et al., 2024)

MICrONS Explorer
https://www.microns-explorer.org/gallery-mm3-renders

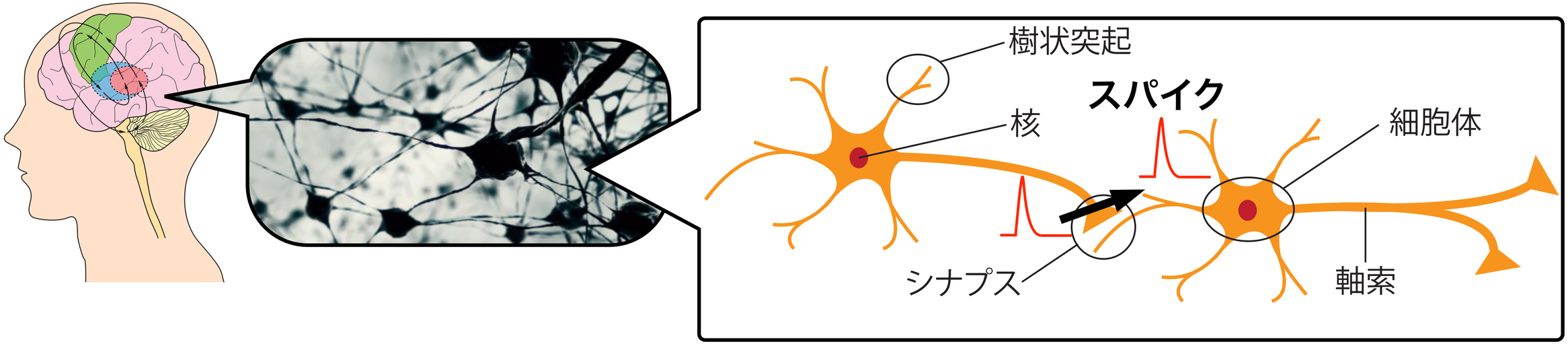
大規模神経回路シミュレーション環境の提供を目指す

目次

- 脳における情報処理
- シミュレーション神経科学
- 細胞形状を考慮する神経回路モデル
- 「Neulite」の開発と「富岳」を用いた高速化事例
- 現状の問題点と将来の展望

脳における情報処理

脳とはニューロンによって構成されるネットワーク

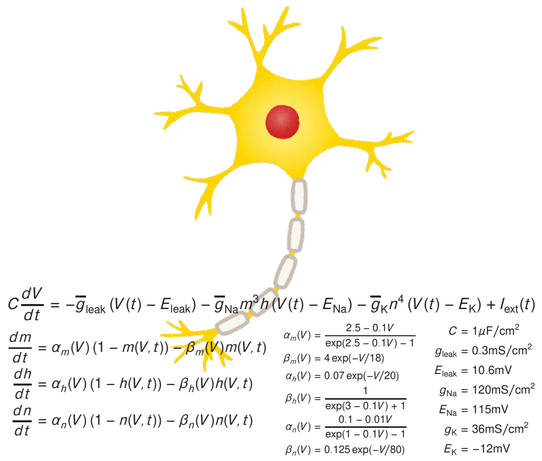


ニューロンが**シナプス**を通じて**スパイク**と呼ばれる電気信号を伝達

ネットワークが機能を生み出す

シミュレーション神経科学

概要



$$C\frac{dV}{dt} = -\overline{g}_{\text{leak}}(V(t) - E_{\text{leak}}) - \overline{g}_{\text{Na}} m^3 h (V(t) - E_{\text{Na}}) - \overline{g}_K n^4 (V(t) - E_K) + I_{\text{ext}}(t)$$

$$\frac{dm}{dt} = \alpha_m(V)(1 - m(V,t)) - \beta_m(V) m(V,t)$$

$$\frac{dh}{dt} = \alpha_h(V)(1 - h(V,t)) - \beta_h(V) h(V,t)$$

$$\frac{dn}{dt} = \alpha_n(V)(1 - n(V,t)) - \beta_n(V) n(V,t)$$

$$\alpha_m(V) = \frac{2.5 - 0.1V}{\exp(2.5 - 0.1V) - 1}$$

$$\beta_m(V) = 4\exp(-V/18)$$

$$\alpha_h(V) = 0.07\exp(-V/20)$$

$$\beta_h(V) = \frac{1}{\exp(3 - 0.1V) + 1}$$

$$\alpha_n(V) = \frac{0.1 - 0.01V}{\exp(1 - 0.1V) - 1}$$

$$\beta_n(V) = 0.125\exp(-V/80)$$

$C = 1\mu\text{F/cm}^2$

$g_{\text{leak}} = 0.3\text{mS/cm}^2$

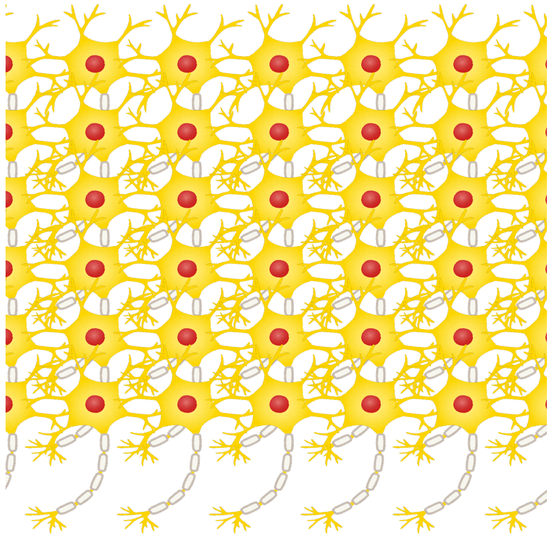
$E_{\text{leak}} = 10.6\text{mV}$

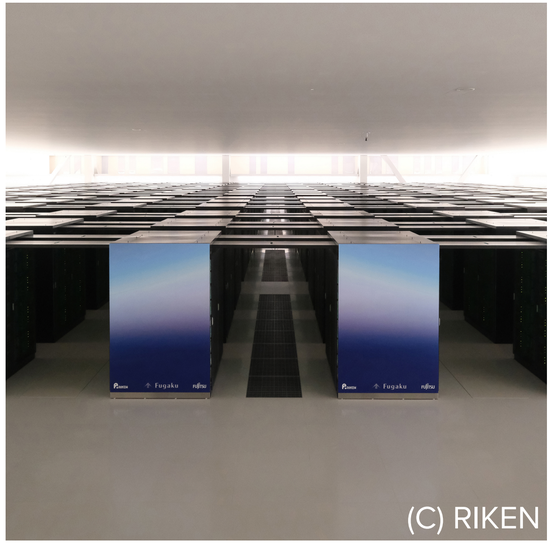
$g_{\text{Na}} = 120\text{mS/cm}^2$

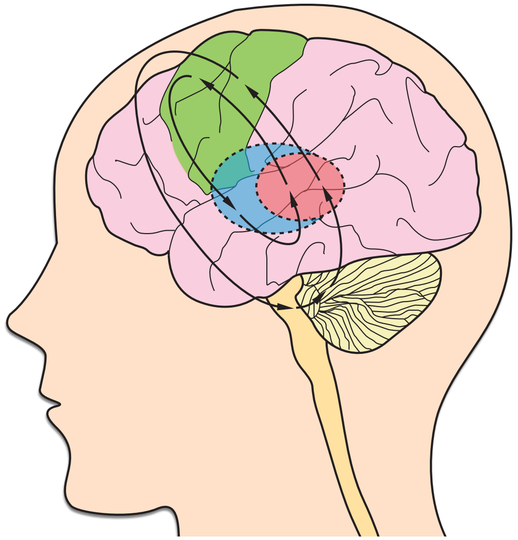
$E_{\text{Na}} = 115\text{mV}$

$g_K = 36\text{mS/cm}^2$

$E_K = -12\text{mV}$







1個のニューロンの
ことは数式で書ける

860億個のニューロン
の数式をプログラム

スパコンで
数値シミュレーション

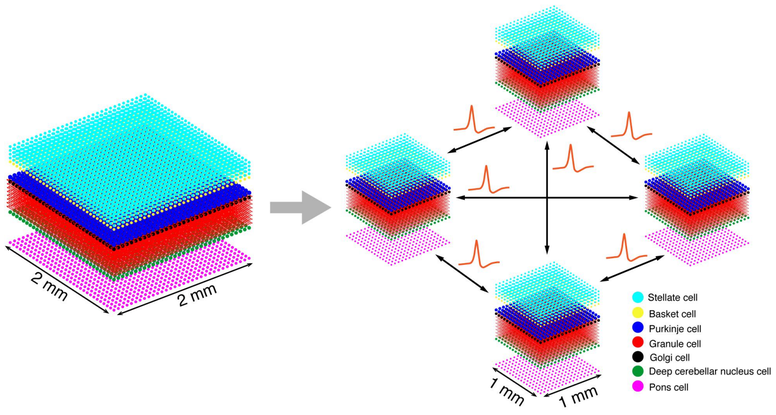
（原理的には）ヒトの
脳の神経活動を再現

操作可能な脳のデジタルコピーを作って、神経活動を再現・予測する

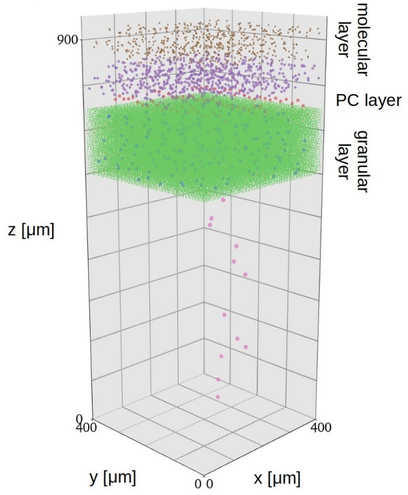
シミュレーション神経科学

これまでの取り組み:ネットワークが機能を生み出すという仮定

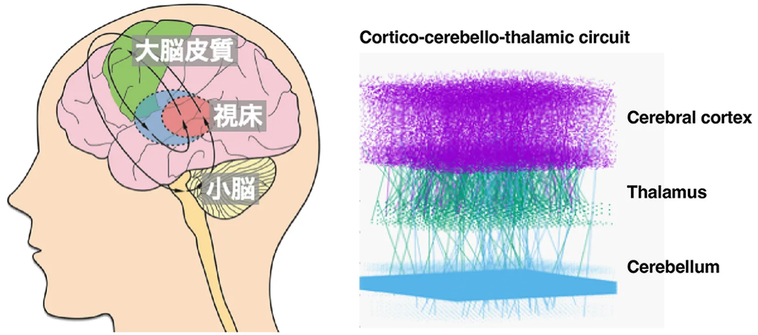
- 1つのニューロンは単純な素子とみなしモデル化
- ネットワークの構築を重視
  「富岳」を用いたヒトスケールシミュレーション (Igarashi et al., 2022)



ヒトスケール小脳 →「京」
(Yamaura et al., 2020)



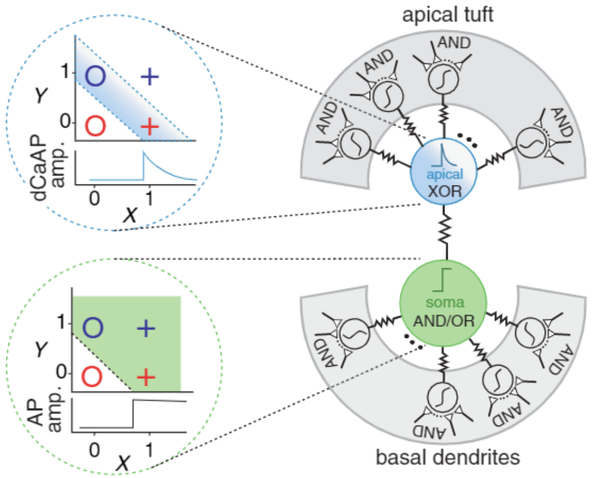
小脳 → GPU
(Kuriyama et al., 2021)

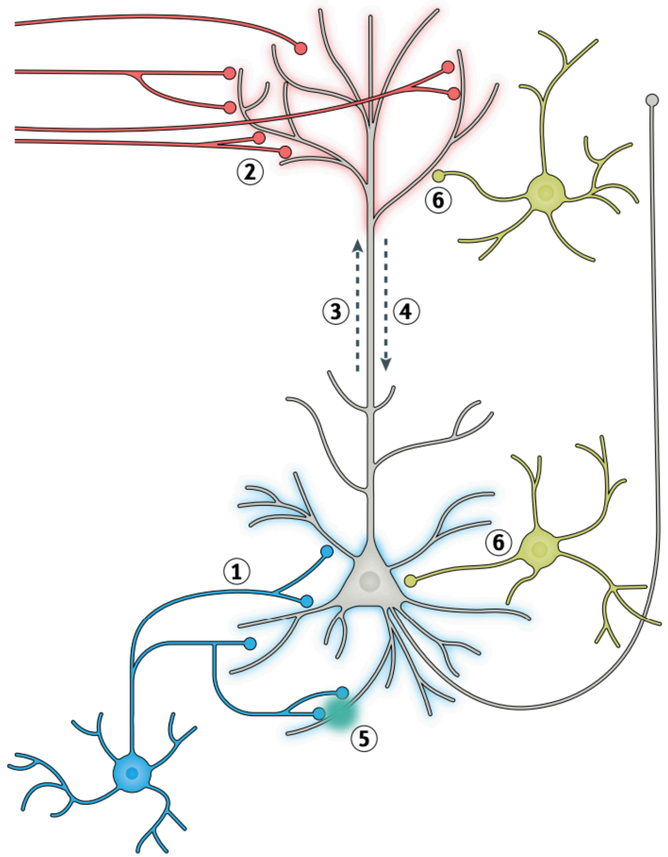


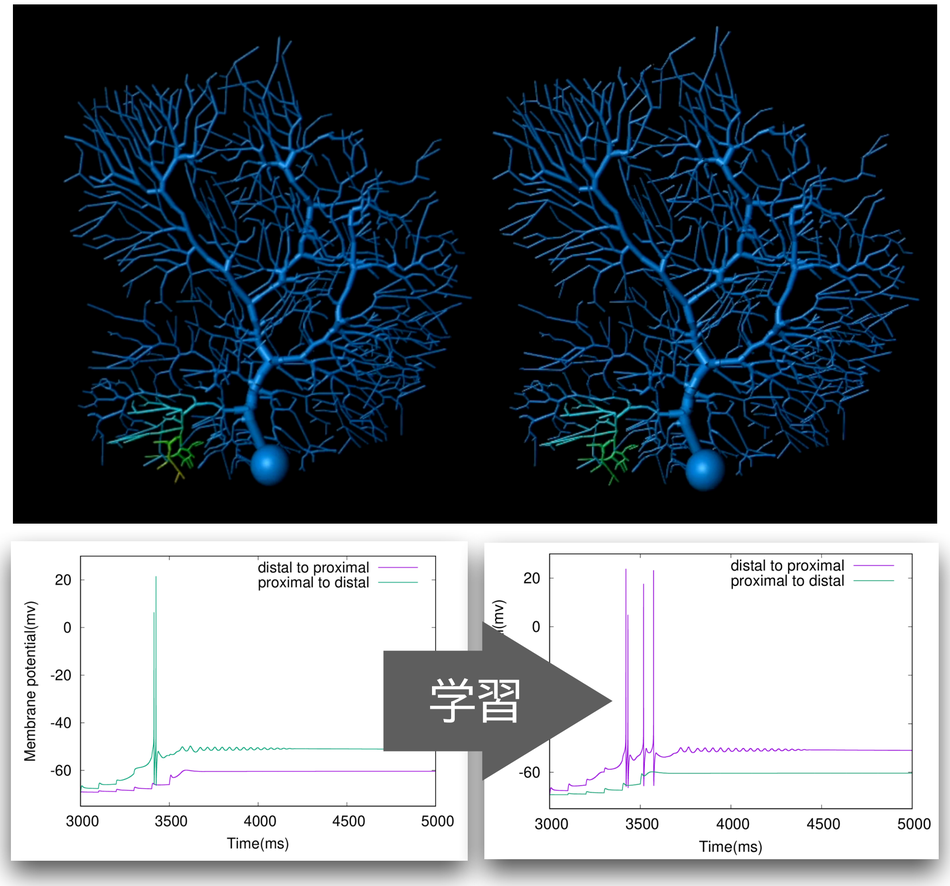
ヒトスケール全脳 →「富岳」
(Igarashi et al., 2022)

シミュレーション神経科学

近年の発見：ニューロン単体でも高度な情報処理を行なっている







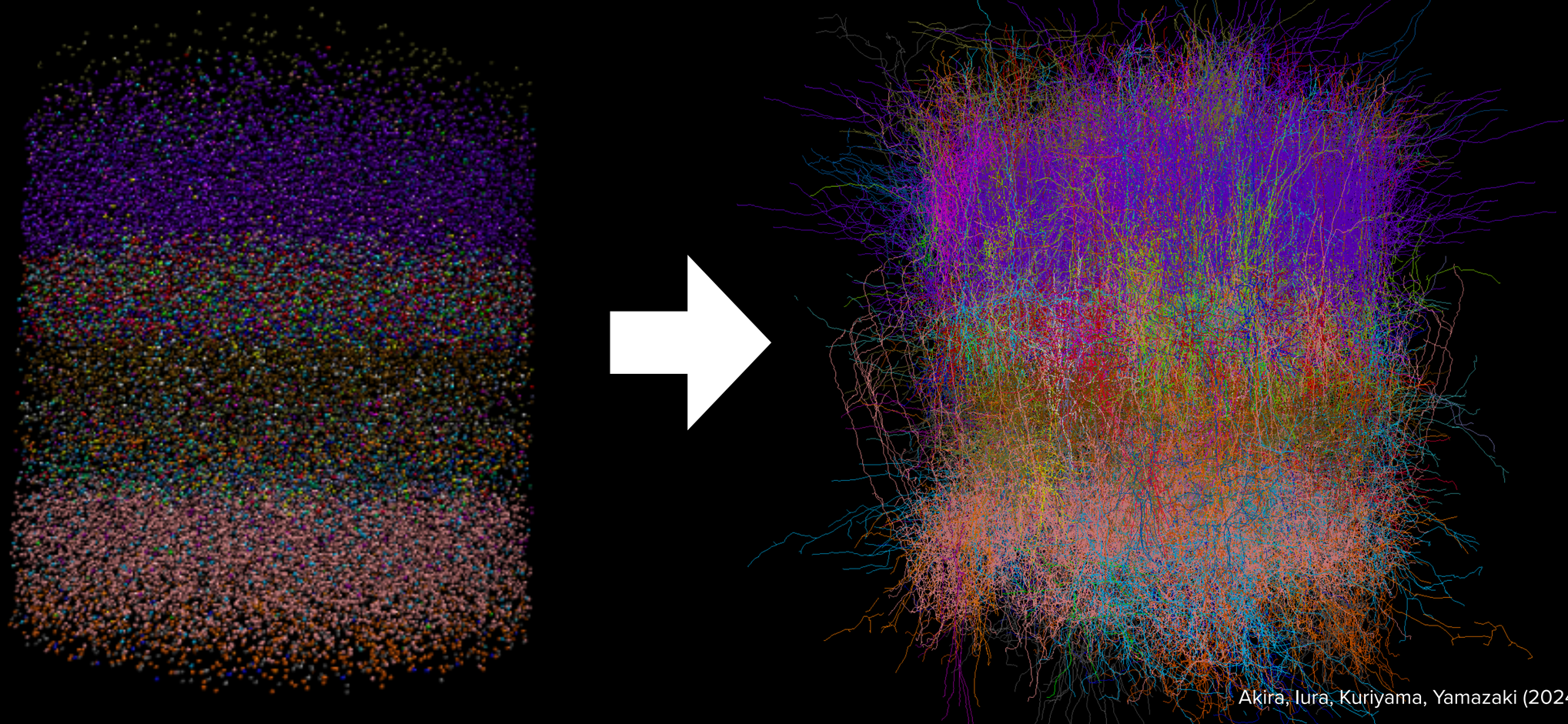
XOR
(Gidon, et al. 2020)

誤差逆伝播
(Lillicrap, et al. 2020)

シーケンス弁別
(Tamura, et al. 2023)

シミュレーション神経科学

問い



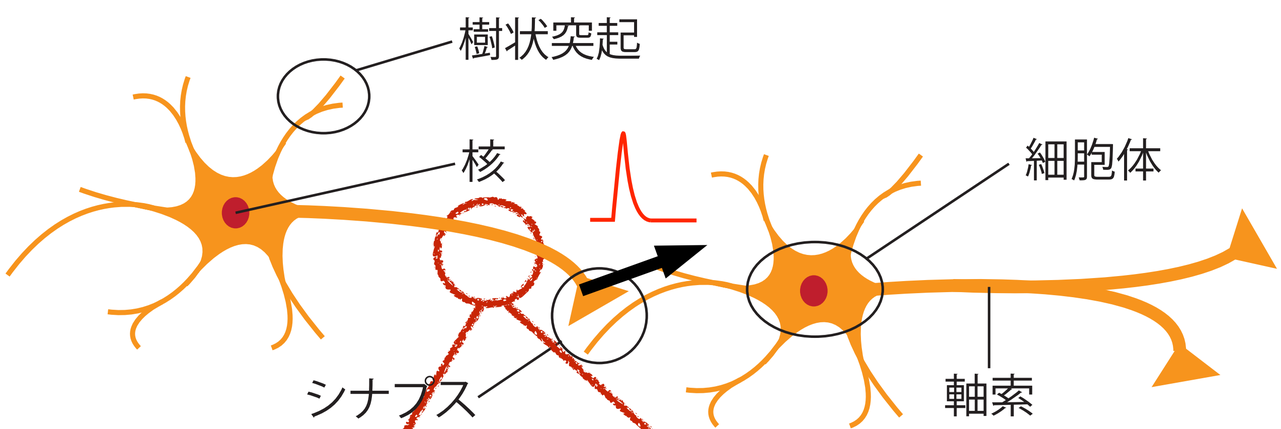
Akira, Iura, Kuriyama, Yamazaki (2024) JNNS2024

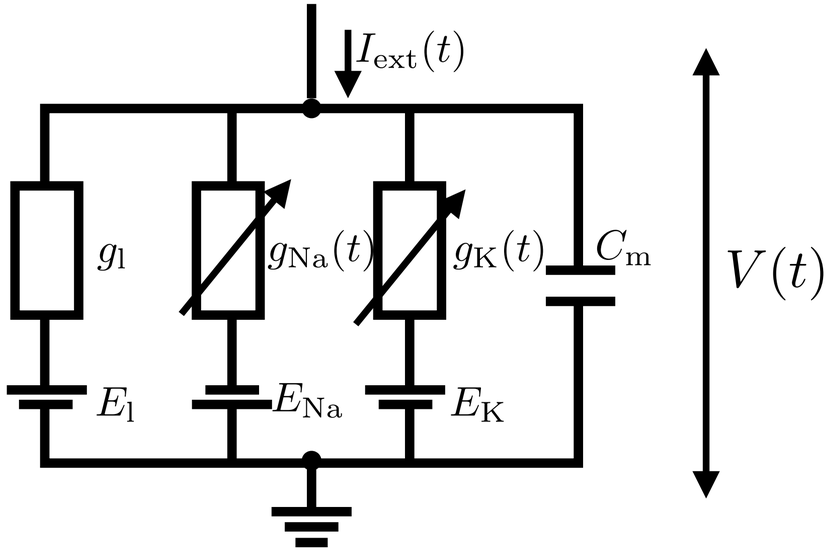
単一ニューロンのミクロな性質はネットワークのマクロな動態にどう影響するか？

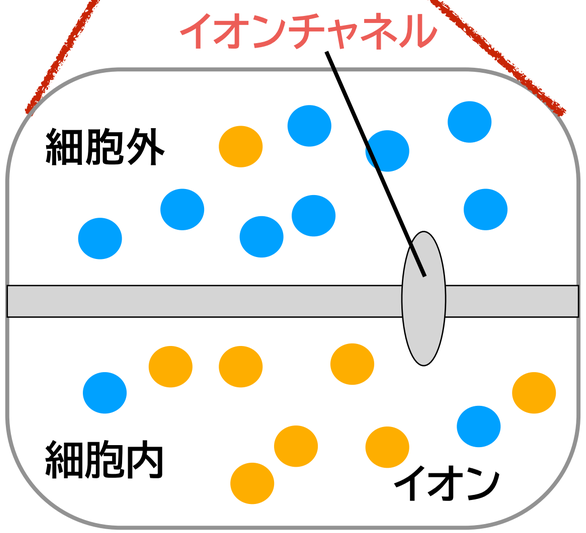
計算量が増加＝スパコンの利用が必須

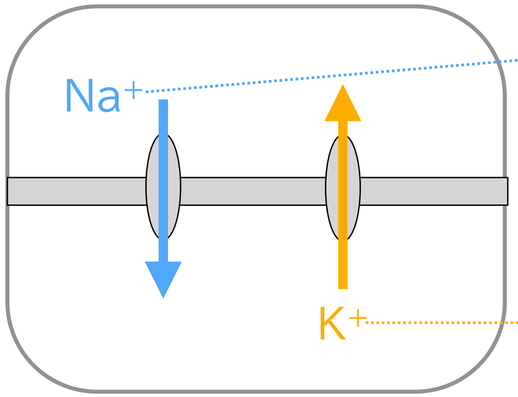
(補足) ニューロンのシミュレーションとは

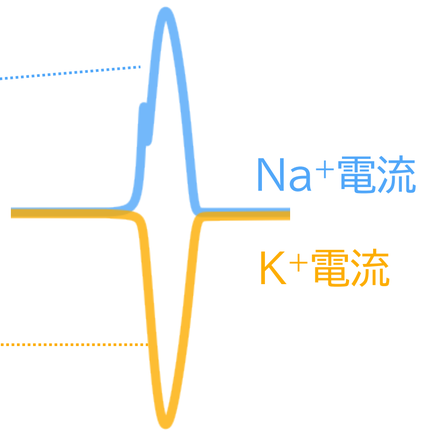
→細胞内外の電位差 = 膜電位を計算し、スパイクを追跡する

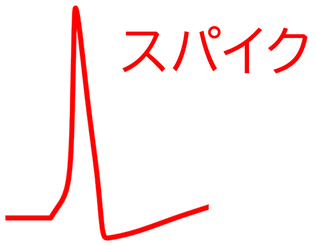




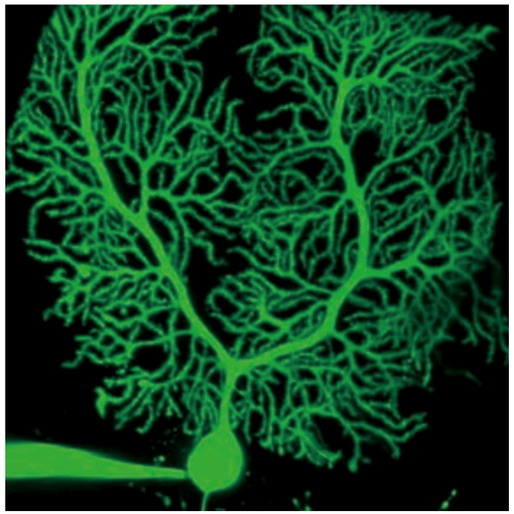


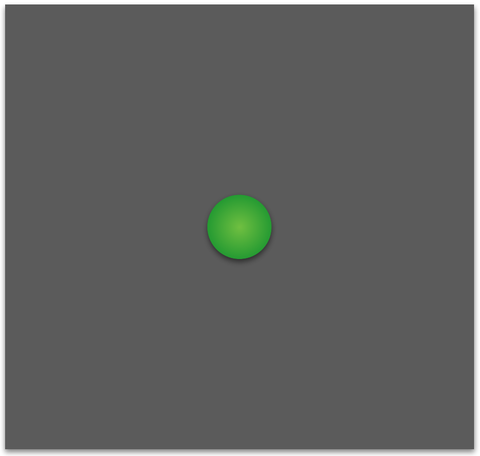






細胞形状を考慮する神経回路モデル
何が違うのか

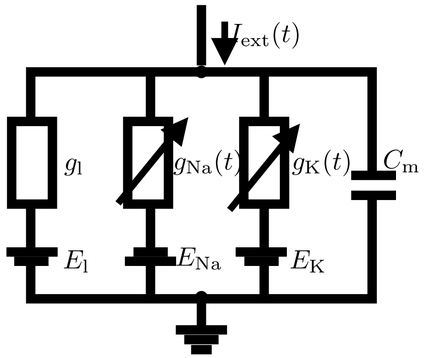


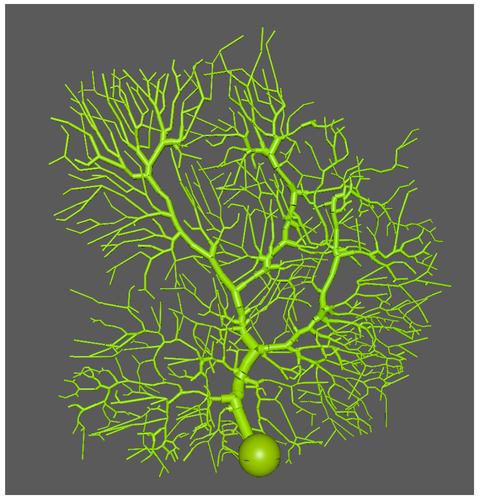


点モデル

一様な膜特性を仮定

常微分方程式を解く



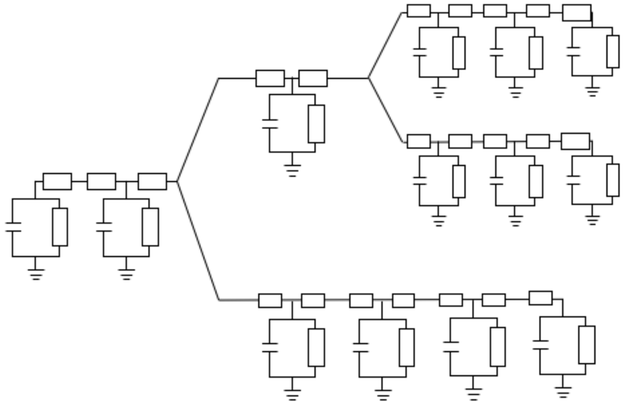


生物物理学的モデル

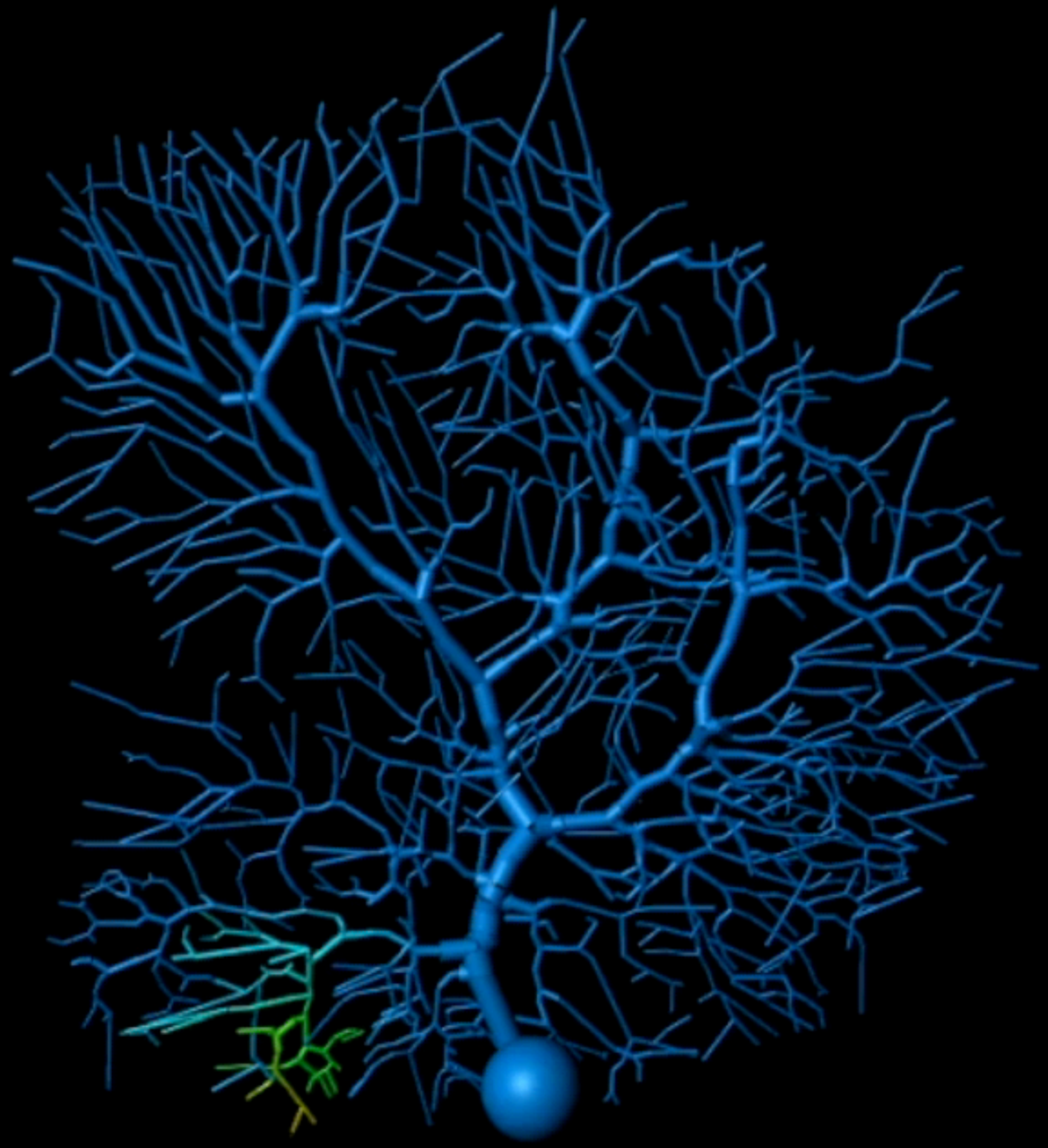
空間的要素

（分岐、チャネル分布）を考慮

偏微分方程式を解く



画像：（左）『理研ニュース』2010年6月号（平成22年6月7日発行）（中央下）Kobayashi, et al.（2021）より形態情報を取得して作成

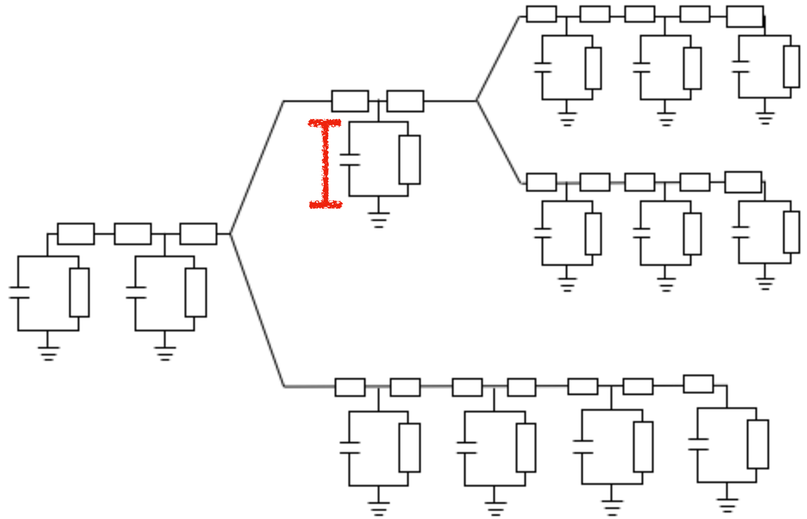


Tamura et al., (2023)

細胞形状を考慮する神経回路モデル

マルチコンパートメントモデル = 反応拡散方程式

$$C_{\mathrm{m}}\frac{dV_j(t)}{dt} = \frac{1}{\pi d l_j}\sum_k \frac{V_k(t)-V_j(t)}{R_{j,k}} - I_{\mathrm{ion},j}(V_j(t),t) + I_{\mathrm{syn},j}(V_j(t),t) + I_{\mathrm{ext},j}(t)$$



反応拡散方程式として記述

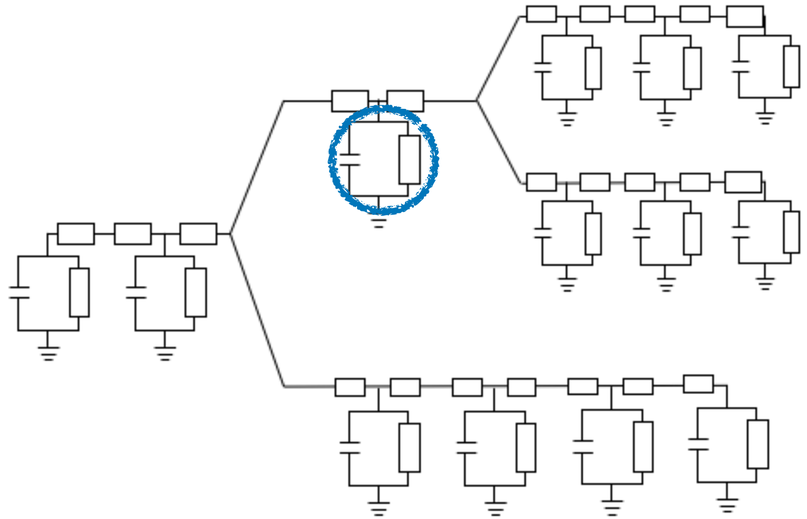
各コンパートメントごとの**膜電位**を計算

- **拡散項**
  - 隣接するコンパートメントへの電位拡散
- **反応項**
  - 分布するチャネルによるイオン電流
  - 他細胞からのシナプス接続によるシナプス電流
  - 対象外の要因からくる外部電流

細胞形状を考慮する神経回路モデル

マルチコンパートメントモデルの数値計算

$$C_{\mathrm{m}}\frac{dV_j(t)}{dt} = \frac{1}{\pi d l_j}\sum_k \frac{V_k(t)-V_j(t)}{R_{j,k}} - I_{\mathrm{ion},j}(V_j(t),t) + I_{\mathrm{syn},j}(V_j(t),t) + I_{\mathrm{ext},j}(t)$$



反応項

- **イオン電流**

  チャネルの種類によって実装が異なる

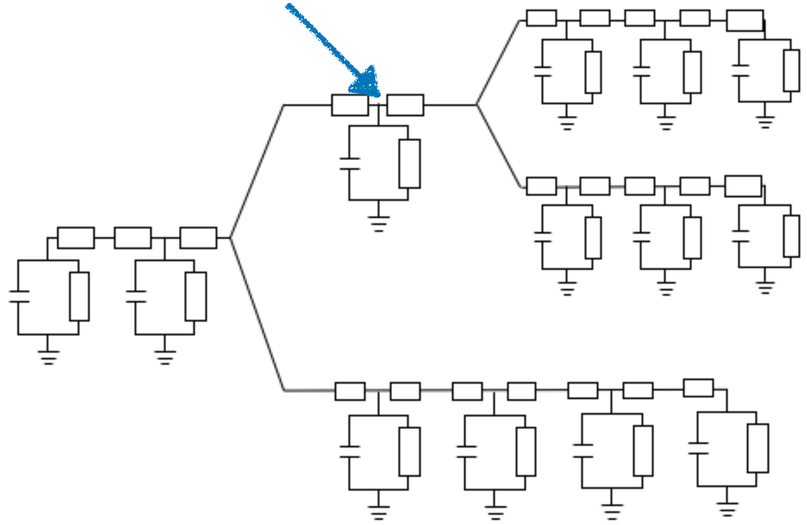
  例) 常微分方程式, 状態遷移方程式

細胞形状を考慮する神経回路モデル

マルチコンパートメントモデルの数値計算

$$C_{\mathrm{m}}\frac{dV_j(t)}{dt} = \frac{1}{\pi d l_j}\sum_k \frac{V_k(t)-V_j(t)}{R_{j,k}} - I_{\mathrm{ion},j}(V_j(t),t) + I_{\mathrm{syn},j}(V_j(t),t) + I_{\mathrm{ext},j}(t)$$

他のニューロンからの入力



反応項

- **シナプス電流**

  該当コンパートメントに接続する他ニューロンの
  スパイク発火によって誘発される電流
  → ニューロン同士の接続情報 (重み付き有向グラフ)
  → ネットワークのシミュレーションになる
  ＝ ニューロン間のビット送受信に相当

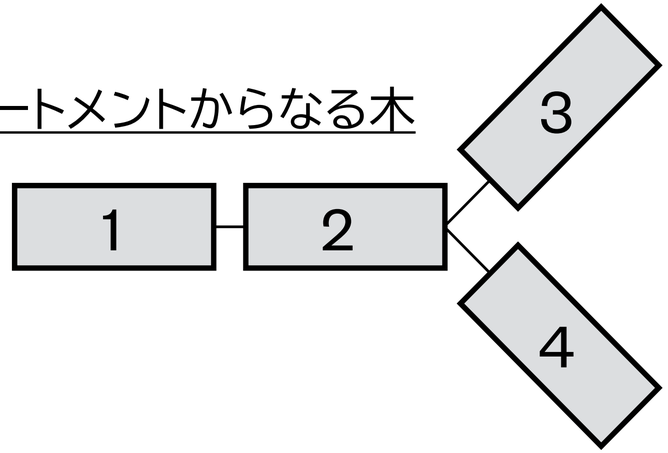
$$I_{\mathrm{syn},j}(V_j(t),t) = \sum_k w_k g_k(t)(V_j(t)-E_k)$$

細胞形状を考慮する神経回路モデル

マルチコンパートメントモデルの数値計算

$$C_{\mathrm{m}}\frac{dV_j(t)}{dt} = \frac{1}{\pi d l_j}\sum_k \frac{V_k(t)-V_j(t)}{R_{j,k}} - I_{\mathrm{ion},j}(V_j(t),t) + I_{\mathrm{syn},j}(V_j(t),t) + I_{\mathrm{ext},j}(t)$$

4コンパートメントからなる木



拡散項と方程式の離散化

- 隣接コンパートメントへ電流が流れる
- **陰解法**：Crank-Nicolson法による離散化が一般的
  コンパートメント数Nに対し、木構造であるため
  行列は疎

$$\begin{bmatrix} V_1(t-\Delta t)+D \\ \vdots \\ V_4(t-\Delta t)+D \end{bmatrix} = \begin{bmatrix} (1+\sigma+\gamma) & -\sigma & 0 & 0 \\ -\sigma & (1+3\sigma+\gamma) & -\sigma & -\sigma \\ 0 & -\sigma & (1+\sigma+\gamma) & 0 \\ 0 & -\sigma & 0 & (1+\sigma+\gamma) \end{bmatrix} \begin{bmatrix} V_1(t) \\ \vdots \\ V_4(t) \end{bmatrix}$$

非三重対角行列からなる連立方程式を解く

細胞形状を考慮する神経回路モデル

シミュレーションの構成

各コンパートメント
イオンチャネルの計算 (常微分方程式, マルコフモデルによる状態遷移方程式)

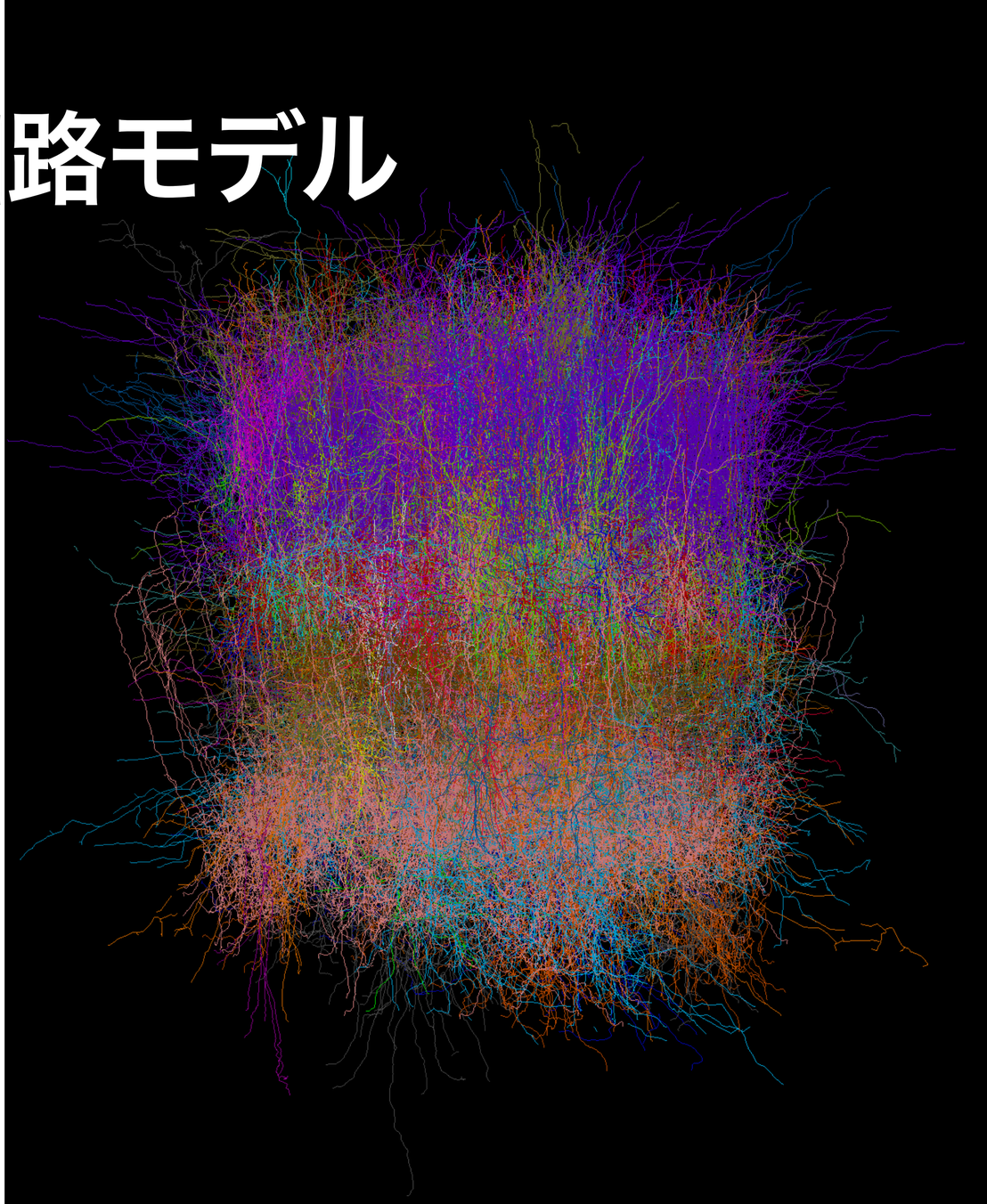
各ニューロン
反応拡散方程式による全コンパートメントの膜電位計算
連立方程式を解く
ニューロンごとに行列は異なる

ネットワーク
スパイク検知→シナプス後細胞へスパイク伝播→シナプスごとに電流を計算

細胞形状を考慮する神経回路モデル

大規模化に向けた技術的課題

- **Allen V1モデル** (Billeh et al., 2023)
  マウス1次視覚野の一部
  約5万ニューロン＋約1600万シナプス
  更なる大規模神経回路には
  スパコンの利用が不可欠
- NEURONシミュレータ
  機能が豊富な代わりにコードが煩雑
  HPC向けの最適化が困難
  「富岳」上でmakeできない
- **「富岳」で動作するマルチコンパートメントモデル用シミュレータが必要**



A light-weight neuron simulator
Neulite の開発

既存の標準化（API、データベース）に従った実装であること

軽量かつ高い拡張性を持つこと（＝最低限の依存関係）

大規模なネットワークのシミュレーションが可能であること

Neulite の開発
既存の標準化 = Brain Modeling Toolkit

Brain Modeling Toolkit（BMTK）

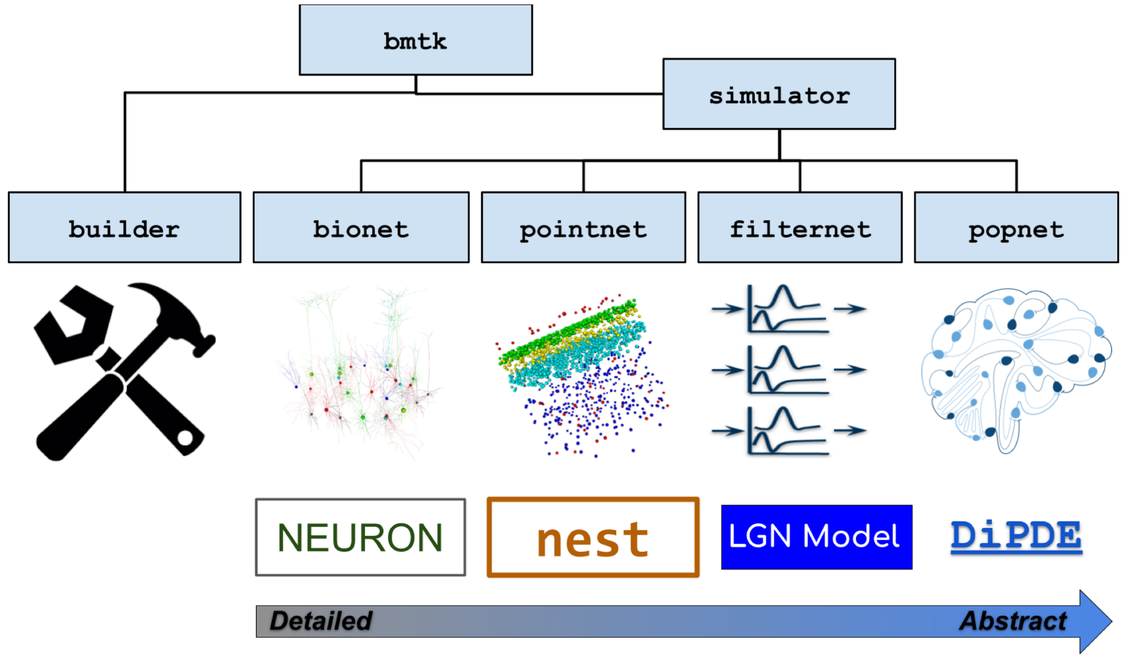
Allen研究所が開発（Dai, et al. 2020）

NEURONを含む複数のシミュレータの実行・結果の統合が可能

Pythonで実装

BioNet（Gratiy, et al. 2018）

NEURONシミュレータを実行するためのPythonインターフェース



BMTKの全体像

Neulite の開発
Brain Modeling Toolkit の利点

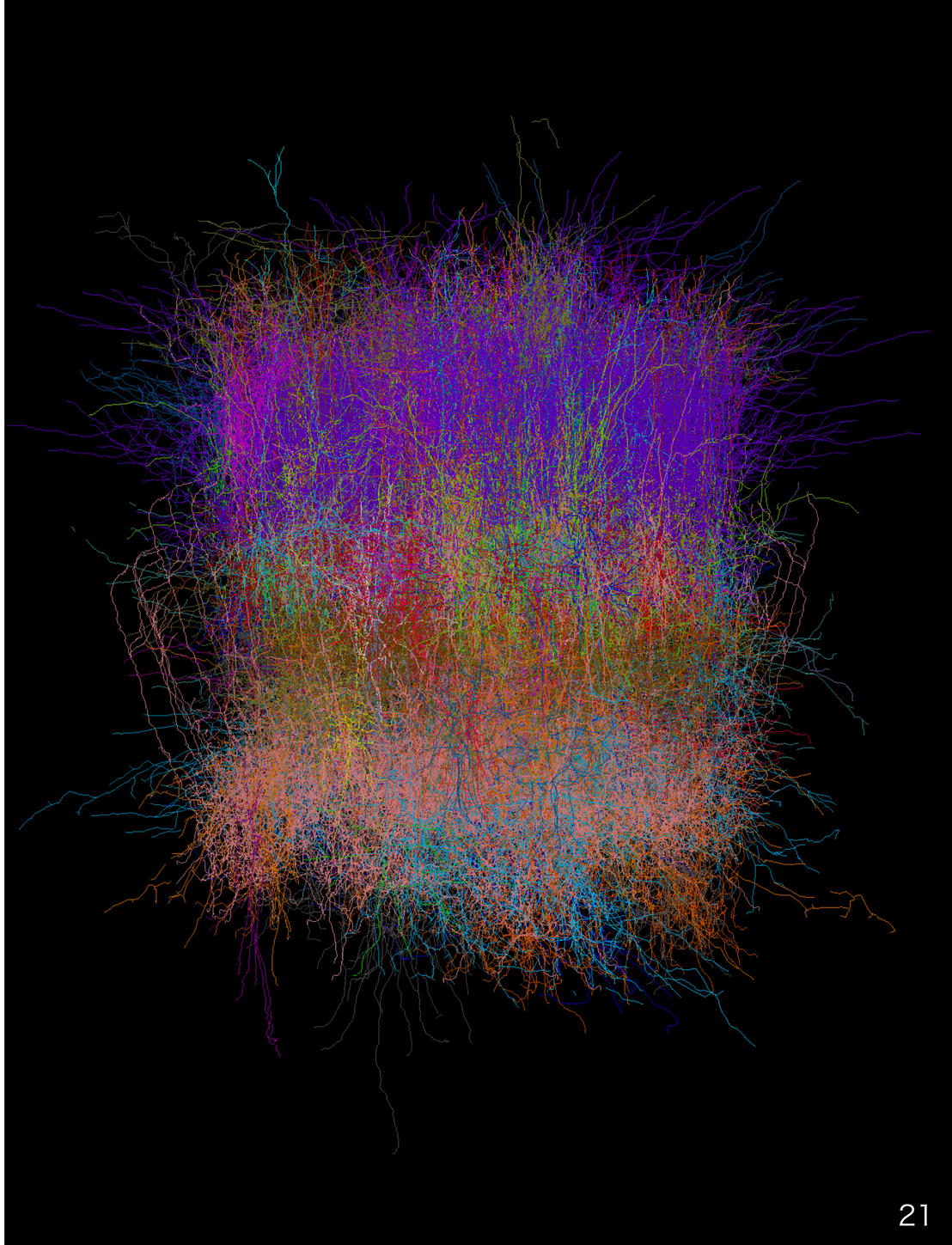
モデリングとシミュレーションのためのAPIが用意されている

利用するニューロンデータが標準化されている
(Allen Cell-Type Database, **CTDB**)

イオンチャネルが限定 (15種)

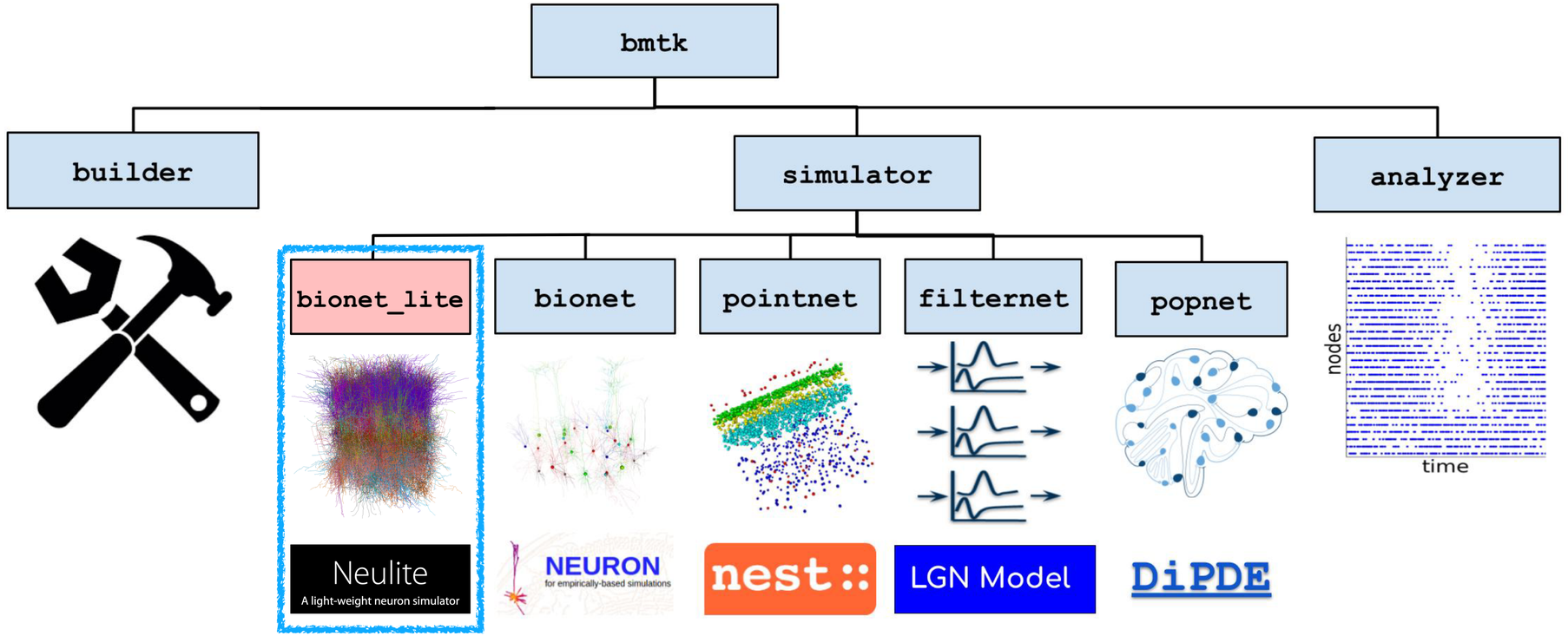
標準に従うことで開発が容易になる
→シミュレータの軽量化

CTDB の全ての神経細胞を扱える



Neulite の開発

Neulite の位置付け：新たな大規模シミュレーション向け simulator



- フロントエンド：入力データ作成
  ニューロン集団定義, ニューロン間接続定義, Cヘッダファイル

- **カーネル：実際のシミュレータ = Neulite**

Neulite の開発

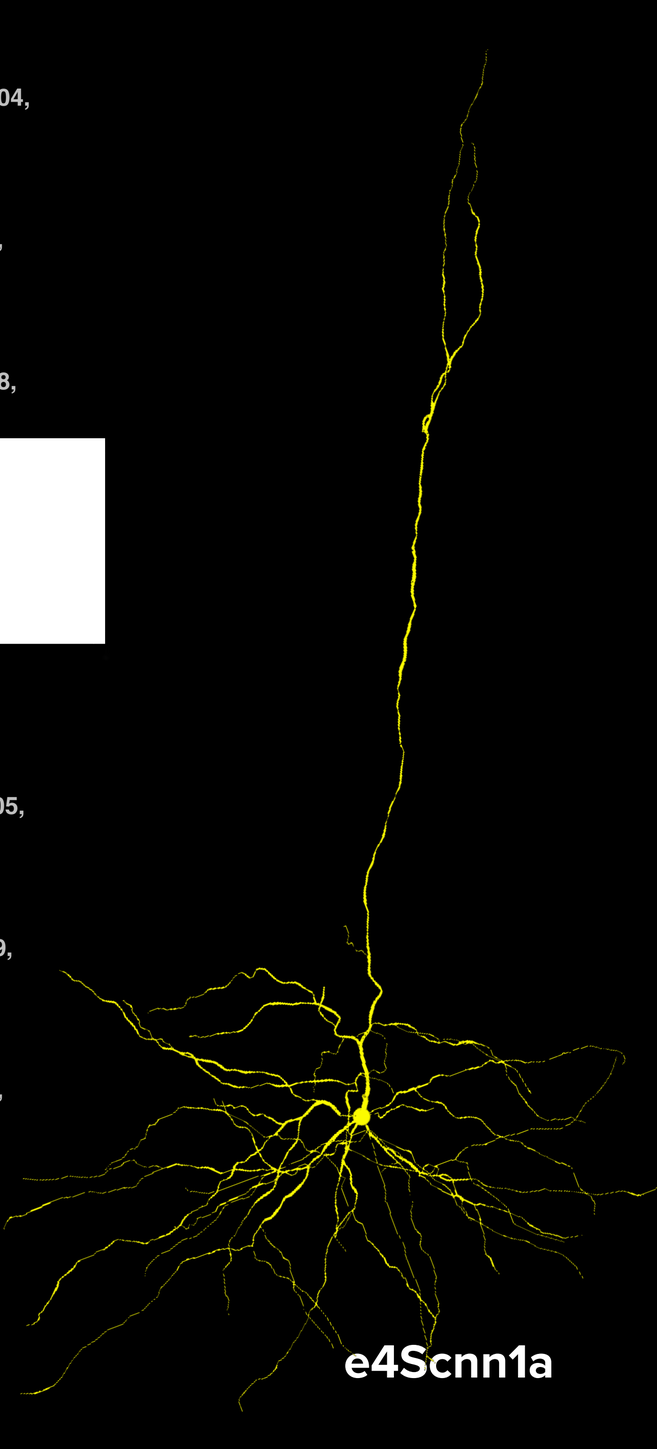
CTDBに従う数理モデルと数値計算法

マルチコンパートメントモデル

$$C_{\mathrm{m}}\frac{dV_j(t)}{dt} = \frac{1}{\pi d l_j}\sum_k \frac{V_k(t)-V_j(t)}{R_{j,k}} - I_{\mathrm{ion},j}(V_j(t),t) + I_{\mathrm{syn},j}(V_j(t),t) + I_{\mathrm{ext},j}(t)$$

- **拡散項**
  - 隣接コンパートメントへの拡散
- **反応項**
  - イオン電流
  - シナプス電流


```
],
"genome": [
 {
  "value": 0.00030357031297000004,
  "section": "soma",
  "name": "gbar_Ih",
  "mechanism": "Ih"
 },
 {
  "value": 0.052161472289300001,
  "section": "soma",
  "name": "gbar_NaV",
  "mechanism": "NaV"
 },
 {
  "value": 0.0033126476739899998,
  "section": "soma",
  "name": "gbar_Kd",
 {
  "value": 1.2128893375100001,
  "section": "soma",
  "name": "gbar_Kv3_1",
  "mechanism": "Kv3_1"
 },
 {
  "value": 1.4016010650299999e-05,
  "section": "soma",
  "name": "gbar_K_T",
  "mechanism": "K_T"
 },
 {
  "value": 0.0011153668151199999,
  "section": "soma",
  "name": "gbar_Im_v2",
  "mechanism": "Im_v2"
 },
 {
  "value": 0.048152669735999999,
  "section": "soma",
  "name": "gbar_SK",
  "mechanism": "SK"
 },
 {
  "value": 0.0,
  "section": "soma",
  "name": "gbar_Ca_HVA",
  "mechanism": "Ca_HVA"
 },
 {
  "value": 0.0,
  "section": "soma",
  "name": "gbar_Ca_LVA",
  "mechanism": "Ca_LVA"
```

Neulite の開発

CTDBに従う数理モデルと数値計算法

マルチコンパートメントモデル

$$C_{\mathrm{m}}\frac{dV_j(t)}{dt} = \frac{1}{\pi d l_j}\sum_k \frac{V_k(t)-V_j(t)}{R_{j,k}} - I_{\mathrm{ion},j}(V_j(t),t) + I_{\mathrm{syn},j}(V_j(t),t) + I_{\mathrm{ext},j}(t)$$

反応項：イオン電流 =14種類のHodgkin-Huxley型チャネル + 1つのMarkov型チャネル

- Hodgkin-Huxley型チャネル

$$I_{\mathrm{ion}}(V(t),t) = \sum_k [\bar{g}_k m^{p_k} h^{q_k}(V - E_k)]$$

m, h について $\quad \dfrac{d}{dt}x(t) = -\dfrac{1}{\tau_x(V)}(x - x_\infty(V))$

$$\tau_x(V) = [\alpha_x(V) + \beta_x(V)]^{-1}, x_\infty(V) = \frac{\alpha_x(V)}{\alpha_x(V) + \beta_x(V)}$$

α_x, β_xは種類ごとに定義される指数関数

カルシウムイオン濃度も各コンパートメントごとに計算

常微分方程式の数値計算には cnexp 法を用いる

$$x_{t+\Delta t} = x_\infty(V) + (x_t - x_\infty(V))\exp\left(-\frac{\Delta t}{\tau_x(V)}\right)$$

Neulite の開発

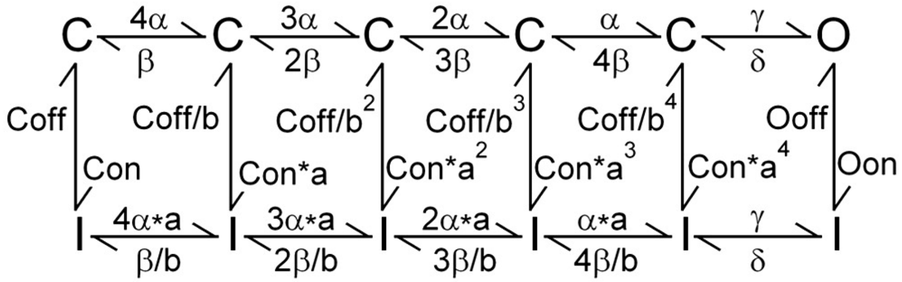
CTDBに従う数理モデルと数値計算法

マルチコンパートメントモデル

$$C_{\mathrm{m}}\frac{dV_j(t)}{dt} = \frac{1}{\pi d l_j}\sum_k \frac{V_k(t)-V_j(t)}{R_{j,k}} - I_{\mathrm{ion},j}(V_j(t),t) + I_{\mathrm{syn},j}(V_j(t),t) + I_{\mathrm{ext},j}(t)$$

反応項：イオン電流 =14種類のHodgkin-Huxley型チャネル + 1つのMarkov型チャネル

- Markov型チャネル (Carter et al., 2012)



C：閉状態, O：開状態, I：不活性化状態

各反応速度が膜電位の関数

イオン電流 = $\bar{g}\cdot O(t)\cdot(V(t)-E_{\mathrm{rev}})$

一次連立微分方程式を解く

現在はガウスの消去法を利用

Neulite の開発

CTDBに従う数理モデルと数値計算法

マルチコンパートメントモデル

$$C_{\mathrm{m}}\frac{dV_j(t)}{dt} = \frac{1}{\pi d l_j}\sum_k \frac{V_k(t)-V_j(t)}{R_{j,k}} - I_{\mathrm{ion},j}(V_j(t),t) + I_{\mathrm{syn},j}(V_j(t),t) + I_{\mathrm{ext},j}(t)$$

反応項：シナプス電流

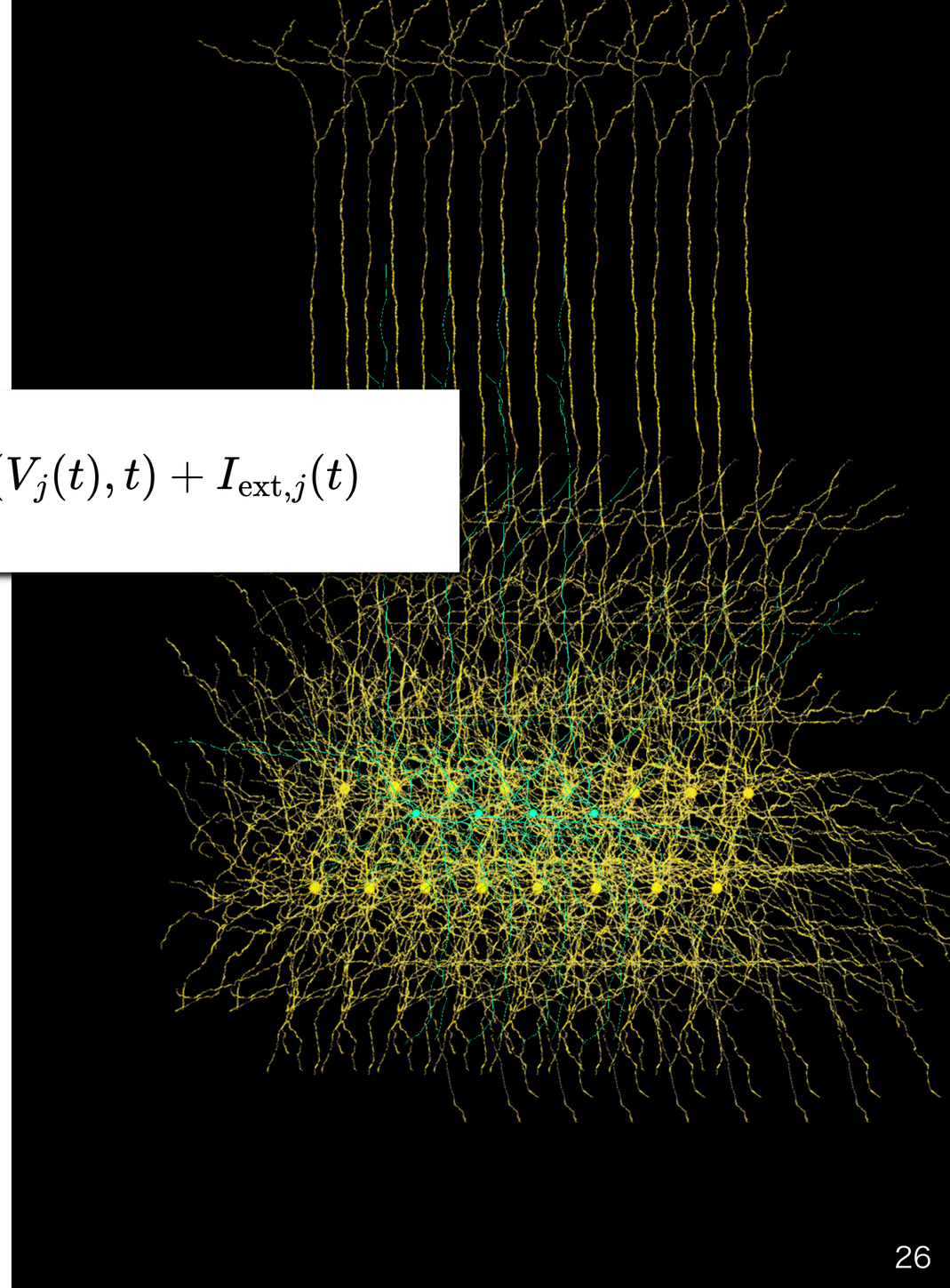
- コンダクタンスベース指数減衰シナプス

$$I_{\mathrm{syn},j}\left(V_j(t),t\right) = \sum_k w_k g_k(t)\left(V_j(t)-E_k\right)$$

$$\frac{d}{dt}g_k(t) = -\frac{1}{\tau_k}g_k(t) + \sum_f \delta_D\left(t-t_k^f\right)$$

$$g_{k_{t+\Delta t}} = g_{k_t}\exp\left(-\frac{\Delta t}{\tau_k}\right) + S_k(t)$$

$$S_k(t) = \begin{cases} 1 & \text{発火} \\ 0 & \text{otherwise} \end{cases}$$



Neulite の開発

CTDBに従う数理モデルと数値計算法

マルチコンパートメントモデル

$$C_{\mathrm{m}}\frac{dV_j(t)}{dt} = \frac{1}{\pi d l_j}\sum_k \frac{V_k(t)-V_j(t)}{R_{j,k}} - I_{\mathrm{ion},j}(V_j(t),t) + I_{\mathrm{syn},j}(V_j(t),t) + I_{\mathrm{ext},j}(t)$$

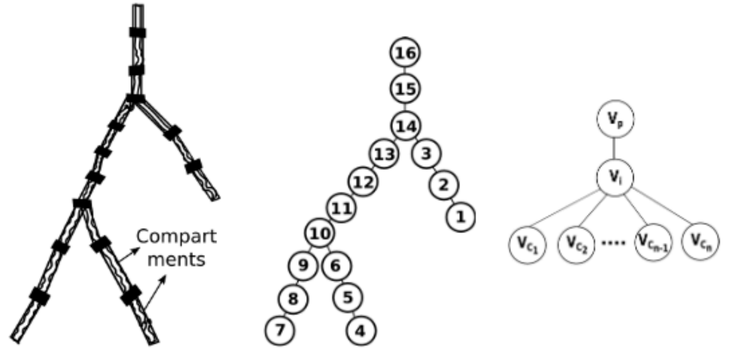
反応拡散方程式の数値計算：Hines' method (Hines, 1984)

- 刻み幅 $\frac{\Delta t}{2}$ による後退オイラー法による離散化から$V\left(t+\frac{\Delta t}{2}\right)$を求め$V(t+\Delta t) = 2V\left(t+\frac{\Delta t}{2}\right) - V(t)$とする
- 時間精度に関して Crank-Nicolson 法と等価 (Hines, 1984)
- Hines' method
  - 枝分かれした木の反応拡散方程式を, コンパートメントの番号づけを工夫して効率的に解く手法
  - 三重対角行列を解く Thomas method に近い思想

Neulite の開発

Hines' method (Hines, 1984)

- 形態
  - コンパートメント間の親子関係 + 抵抗値
  - 疎な対称行列：対角成分+親子間の要素のみが非ゼロ
- 方針
  - 番号づけを工夫することで、直接解法における fill-in を防ぐ
  - オリジナルは深さ優先探索の逆順で番号づけ
  - 前進消去+後退代入により直接解く
  - **親ノードID pid[N] + 対角成分配列 Ad[N] + 親子間要素 Api[N] のみを確保**



(a) Compartmentalized neuron (b) Neuron tree (c) Parent and children of i^{th} compartment

Fig. 1: Multi compartment neuron modelling.

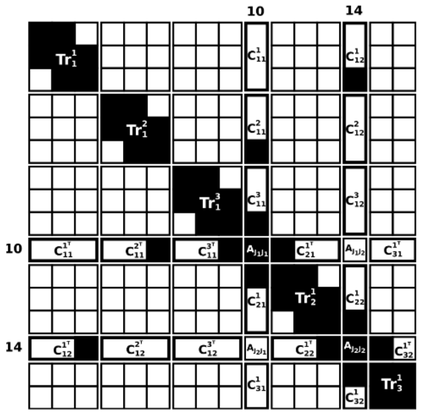


Fig. 2: The general form of Hines matrix corresponding to compartmentalized neuron in Figure 1(b) with J=[10,14].

(Vooturi et al., 2017)

- Neulite における実装
  - 実際の形態情報が保存される .swc (id,type, x,y,z,r,pid) 番号づけに制限がない
  - 細胞体が0番である方がスパイク検知等に都合がいい
    - **前処理：細胞体を0番とする深さ優先探索順にid割り振り**
    - **計算：後退消去+前進代入による直接解法**

Neulite の「富岳」向け高速化

既存の標準化（API、データベース）に従った実装であること

軽量かつ高い拡張性を持つこと（＝最低限の依存関係）

大規模なネットワークのシミュレーションが可能であること

(再掲) 細胞形状を考慮する神経回路モデル
シミュレーションの構成

各コンパートメント
イオンチャネルの計算 (常微分方程式, マルコフモデルによる状態遷移方程式)

各ニューロン
反応拡散方程式による全コンパートメントの膜電位計算
連立方程式を解く
ニューロンごとに行列は異なる

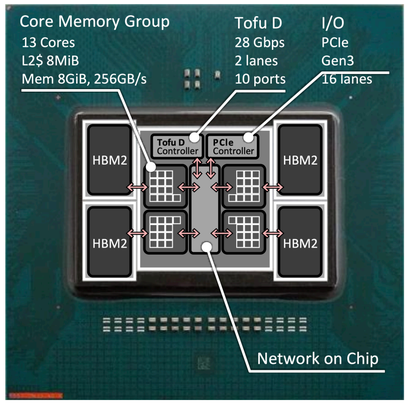
ネットワーク
スパイク検知→シナプス後細胞へスパイク伝播→シナプスごとに電流を計算

Neuliteの「富岳」向け高速化

基本情報

- 総ノード数 158796ノード

A64FX

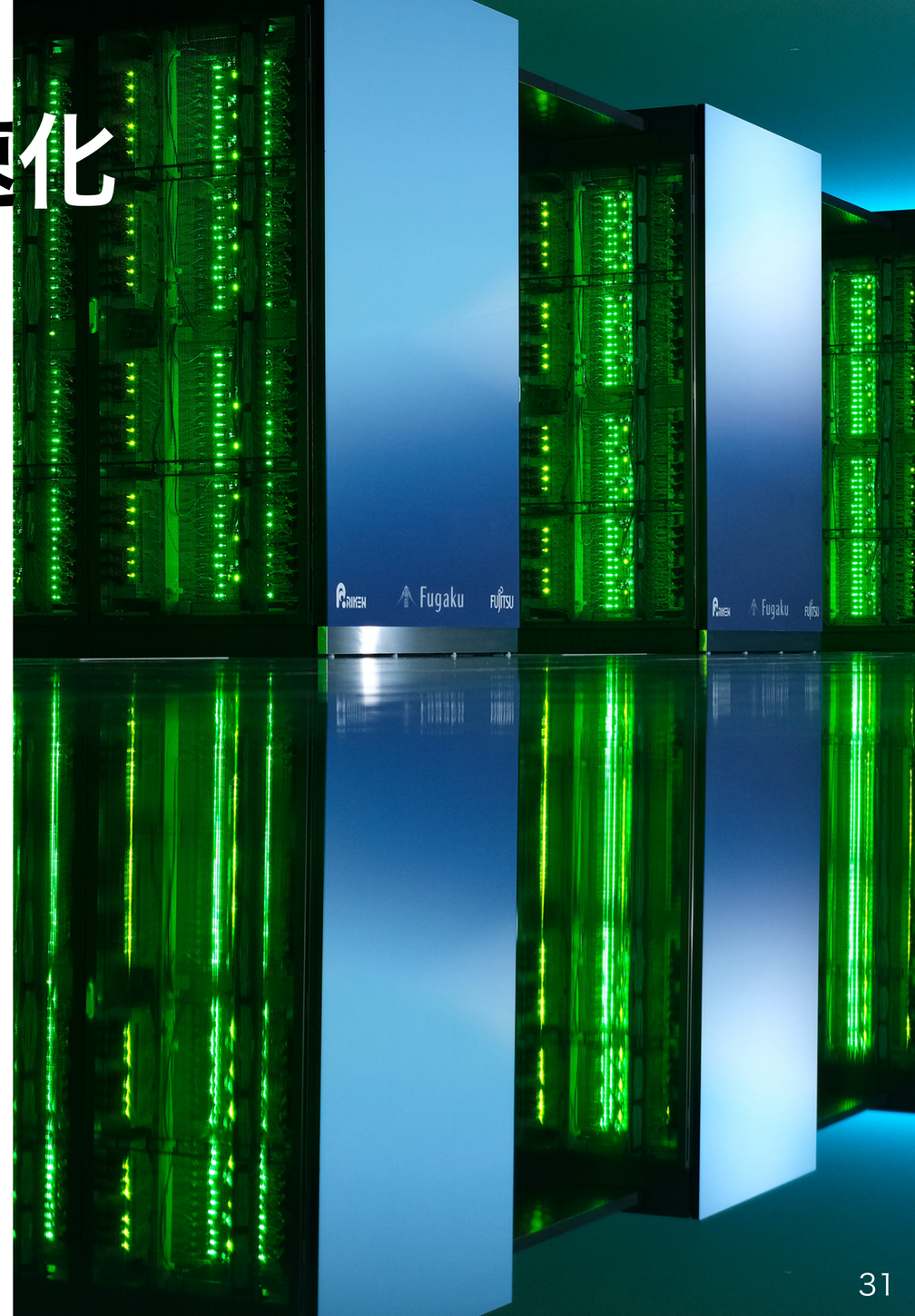


4 CMG / Chip (Node)
12 Cores / CMG
512-bit SIMD FMA
* 2 / Core
Clock 2.0 GHz

Peak performance
- 64 GFLOPS / Core
- Around 3 TFLOPS / Node

- 4 CMG/node : 4プロセス並列
- 12 cores/CMG : 12スレッド並列
- 512-bit SIMD FMA * 2/Core : ベクトル演算

SIMDとFMAを活用することが重要

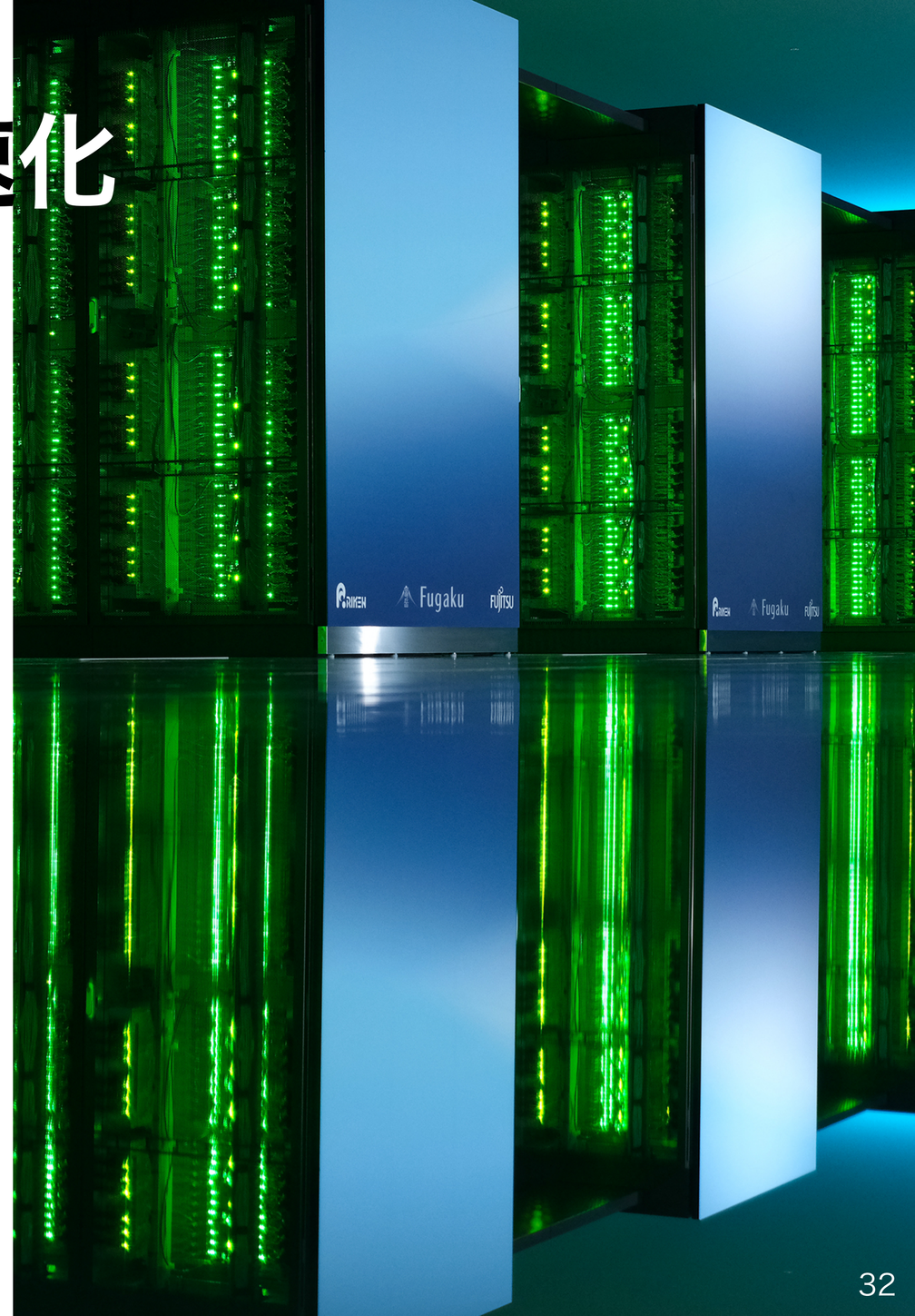


Neulite の「富岳」向け高速化

基本方針

プロセス並列＋スレッド並列＋SIMD化

- プロセス並列 = **神経細胞の分散**
- スレッド並列 = **Hines' Method の並列化**
- SIMD = **複数ニューロンの同時計算**

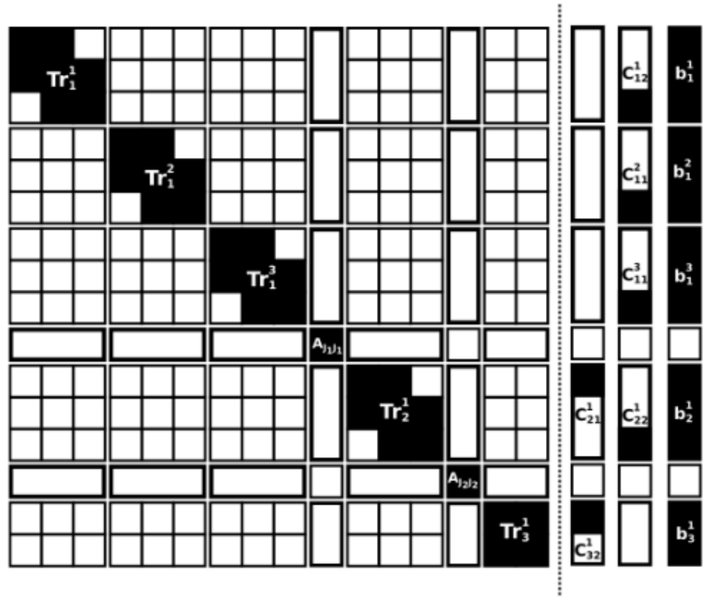


Neulite の「富岳」向け高速化

スレッド並列 : Hines' Method の並列化 (Mascagni, 1991; Hines et al., 2008; Vooturi et al., 2017)

- いわゆる領域分割法
- 木構造をセグメントとジャンクションに分割

1. セグメント(サブ領域)をスレッドに割り振る
    1. セグメント毎に同時に3つの右辺について解く
    2. 補行列MとベクトルM_{rhs}を構築
2. $Mx = M_{\mathrm{rhs}}$を解く
3. 解をもとに全要素の解を計算



(Vooturi et al., 2017)

- GPU版実装 (Vooturi et al., 2017) を参考に実装し、「富岳」に合わせてチューニング
- 明示的な分岐に加えて、セグメント長の最大値Kをヒューリスティックに定めてジャンクションを追加 (FINE decomposition). 今回は K = 45
- 12スレッド並列
    - セグメント数→**スレッド間の合計セクション長の均一化**

Neulite の「富岳」向け高速化

SIMD化：ベクトル演算

- 単一ニューロン計算の高速化 vs **複数ニューロンの同時実行**

  現状の大規模神経回路モデルでは細胞形状を使い回すことが多い → 後者

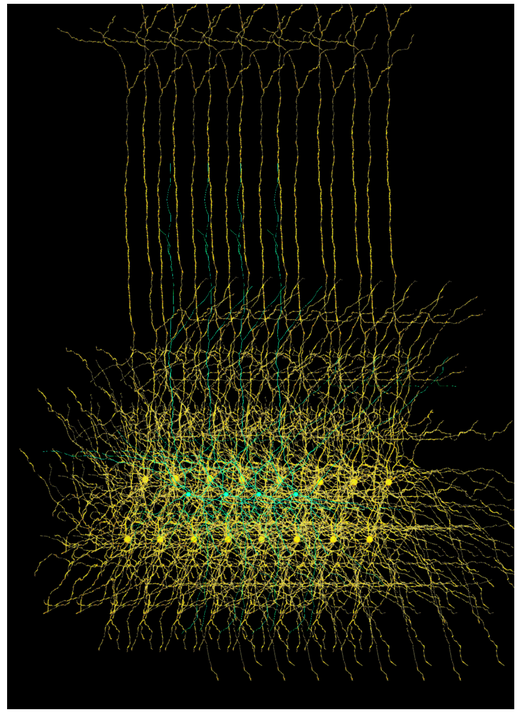
- 512 bit 幅, 倍精度演算 → **同一形態の8ニューロンを同時計算**

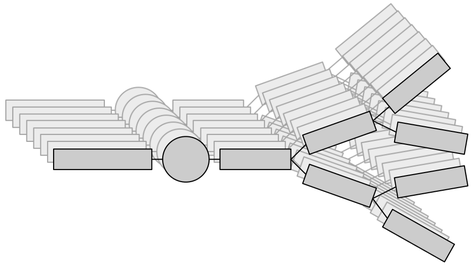
- 汎用的なシミュレータを目指すため計算対象の形状/数が変動

  コンパイル時自動ベクトル化が効きにくい

  → **Arm C Language Extensions for SVE (Scalable Vector Extension)** による明示的なベクトル演算

- Trad ではなく **Clang** でコンパイル
  - Trad の方が指定子等による自動最適化オプションが多い
  - ACLE for SVE 利用時には Clang の方が性能がいい





Neulite の「富岳」向け高速化
富岳向けチューニング

- 割り算, log関数, exp関数の近似関数を作成
- ループのSWP周りがうまく効かない = 必要なループをマクロ関数で定義
  - ループ数決め打ちでもSWPが効かずただ展開するだけ
  - 先の最大セクション長 K を利用し、必要なループをマクロ関数で定義
  - コンパイル時によしなに組み替えてもらう
- アロー演算子等の削減
  - 毎回関数の引数を読み出しアドレス計算を行ってしまう
  - ループブロック毎にローカル変数 `double *arr` に代入しておく
  - predicate mask も、引数に渡したものを使い回すのではなくループブロック毎にローカル変数に渡す

Neulite の「富岳」向け高速化
スパイク通信

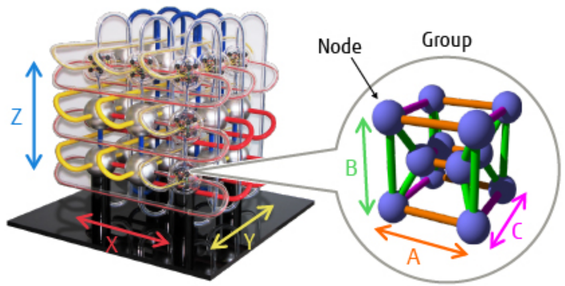
- 元々：各ニューロンスパイクの有無をbitで表現. Allgather で共有
  `spike_arr[nid] = 1 or 0`
- 神経細胞の発火率は最大でも高々数百Hz 程度
  全体で平均するともっと低い

  → 発火したニューロン番号のみを共有する **Sending Indices** 方式
  - 各プロセス内で発火したニューロン数を **Allgather** で共有
    `num_of_spikes[rank] = n, sum(num_of_spikes[]) = total_spiking_neurons`
  - 発火したニューロンIDを並べ、**Allgatherv** で共有
    `global_spiking_neurons [ i ] = nid`
  - 各プロセスにて、`nid` と繋がるシナプス後細胞があればシナプス電流へ足し合わせ

- スパイク通信は 1 ms 毎

Neulite の「富岳」向け高速化
スパイク通信

- Fugakuにおけるチューニング済みアルゴリズムを利用
  - Tofu Unit 単位 (2*3*2ノード) に基づくノード確保
  - デフォルトではメッセージサイズ等を見てアルゴリズムが切り替わる
  - `export OMPI_MCA_coll_select_allgatherv_algorithm=gtvbc` による直接指定

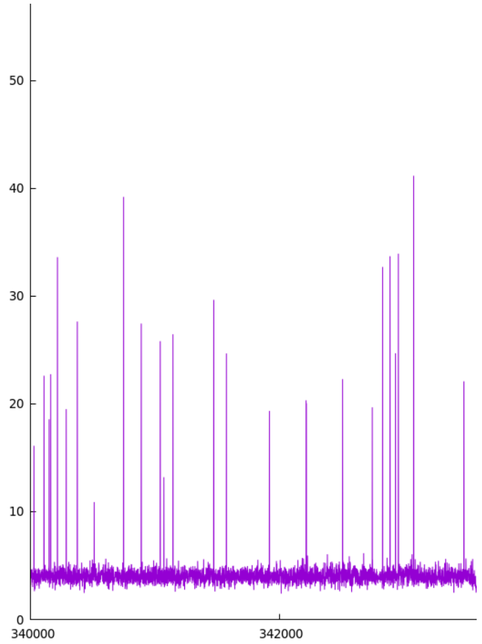


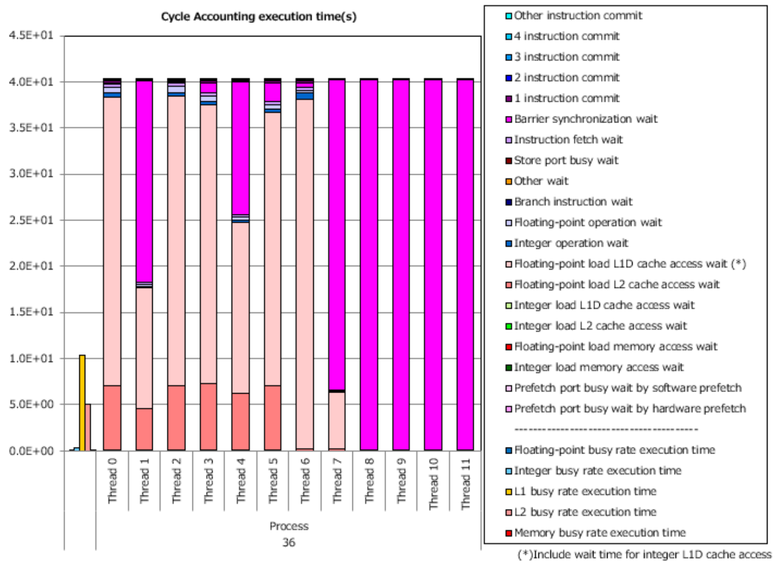

```
--------------- Collective Communication Information --------------
COLLECTIVE            ALG     MSGMIN -     MSGMAX                 NODE       COMM       COUNT AVE.TIME(MSEC)
Allgather             100    1048576 -    4194303       48x      33x    46     291456        1000 1.292
Allgatherv              2          0 -       4095       48x      33x    46     291456           3 6.047
Allgatherv              2       4096 -      16383       48x      33x    46     291456           1 9.073
Allgatherv              2      16384 -      65535       48x      33x    46     291456          77 6.080
Allgatherv              2      65536 -     262143       48x      33x    46     291456         483 8.874
Allgatherv              2     262144 -    1048575       48x      33x    46     291456         263 16.567
Allgatherv            100     262144 -    1048575       48x      33x    46     291456          68 3.767
Allgatherv            100    1048576 -    4194303       48x      33x    46     291456           3 4.507
```

Neulite の「富岳」向け高速化

富岳向けチューニング：実データの取り扱い

- 計算時間が飛び抜けて多いプロセスが発生

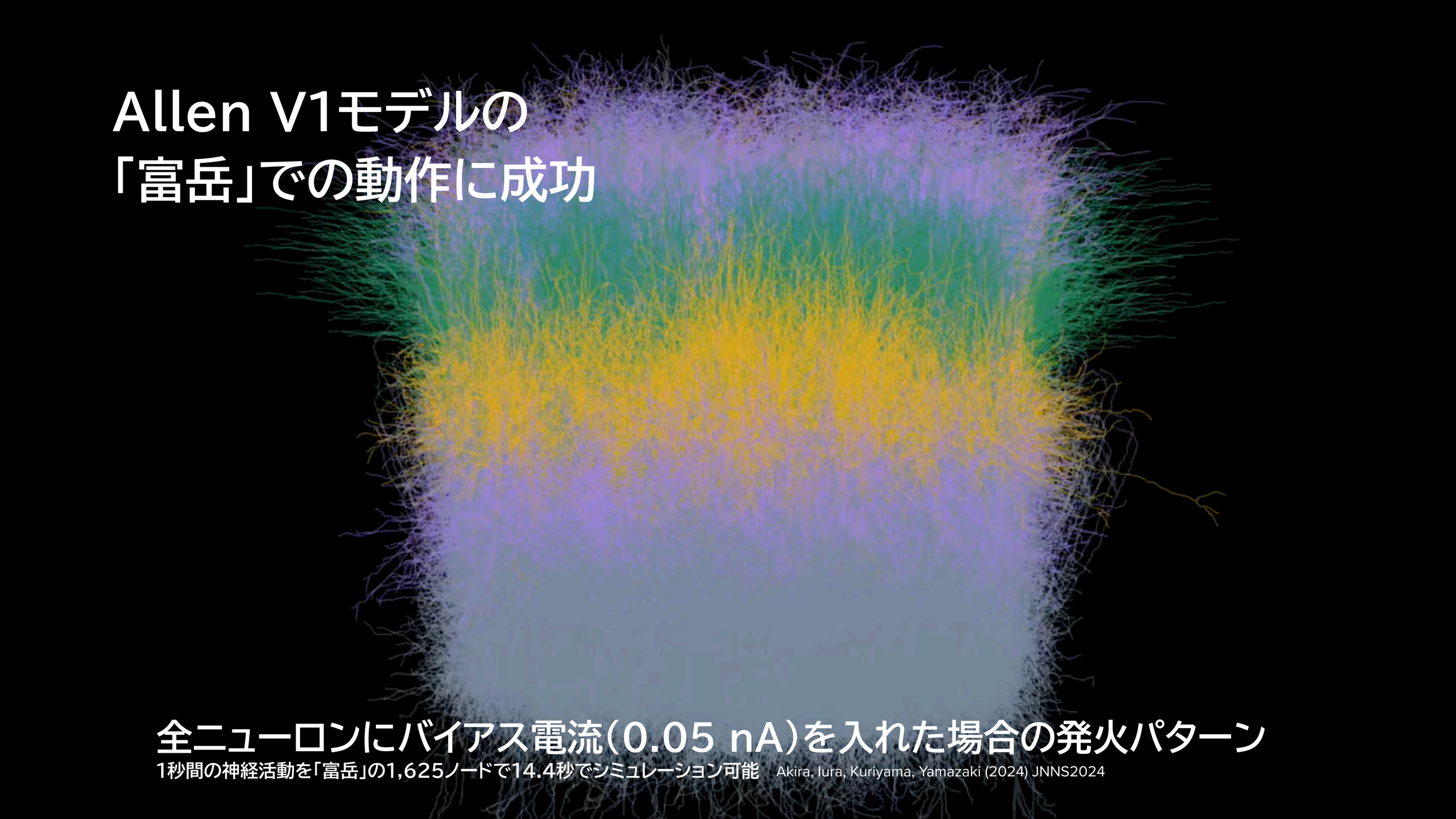




Cache		L1I miss rate (/Effective instruction)	Load-store instruction	L1D miss	L1D miss rate (/Load-store instruction)	L1D miss demand rate (%) (/L1D miss)	L1D miss hardware prefetch rate (%) (/L1D miss)	L1D miss software prefetch rate (%) (/L1D miss)
Process	Thread							
36	0	0.00	5.14E+08	1.59E+08	0.31	91.79%	8.21%	0.01%
36	1	0.00	1.94E+08	6.34E+07	0.33	92.57%	7.43%	0.00%
36	2	0.00	5.55E+08	1.60E+08	0.29	91.15%	8.85%	0.00%
36	3	0.00	4.79E+08	1.51E+08	0.32	91.91%	8.10%	0.00%
36	4	0.00	2.76E+08	9.22E+07	0.33	92.58%	7.43%	0.00%
36	5	0.00	4.62E+08	1.47E+08	0.32	92.00%	8.00%	0.00%
36	6	0.00	4.93E+08	1.55E+08	0.32	91.91%	8.09%	0.00%
36	7	0.00	7.04E+07	1.95E+07	0.28	99.74%	0.29%	-0.03%
36	8	0.01	1.07E+07	1.53E+05	0.01	99.00%	2.41%	-1.41%
36	9	0.01	1.07E+07	1.64E+05	0.02	98.52%	0.68%	0.80%
36	10	0.01	1.08E+07	1.77E+05	0.02	99.06%	1.49%	-0.55%
36	11	0.01	1.07E+07	1.68E+05	0.02	99.15%	1.30%	-0.45%
	CMG 0 total	0.00	3.09E+09	9.48E+08	0.31	92.05%	7.95%	0.00%

- L1D miss 約30%, demand rate も90 %越え = キャッシュスラッシング
- 新規配列の挿入 → 大幅に改善

Allen V1モデルの「富岳」での動作に成功



全ニューロンにバイアス電流（0.05 nA）を入れた場合の発火パターン

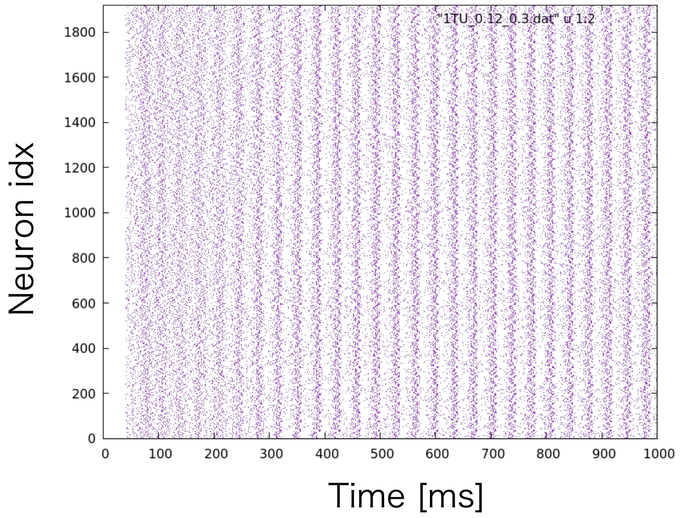
1秒間の神経活動を「富岳」の1,625ノードで14.4秒でシミュレーション可能　Akira, Iura, Kuriyama, Yamazaki (2024) JNNS2024

Neulite の「富岳」向け高速化

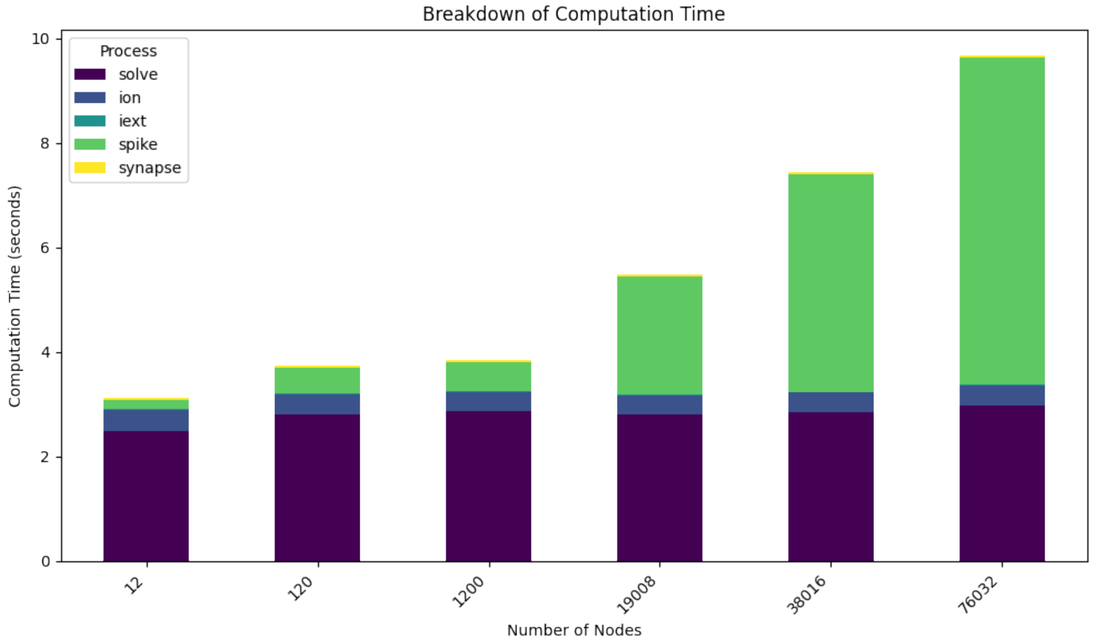
スケーリング性能：バランスネットワーク

- 160 neurons/node. 1ニューロンへ40ニューロンが入力
- dt = 0.1 ms, 1000 ms のシミュレーション

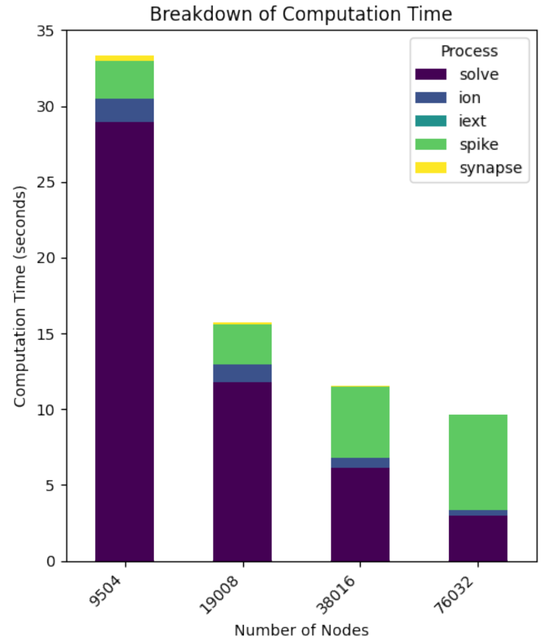
ラスタープロット



弱スケーリング



強スケーリング



- 計算処理部のスケーリングは良いが、通信部が増加.

現状の課題と将来の展望

現状の課題

- 鋭意開発継続中
  - **通信時間の隠蔽**
    - 通信時間の増加が大きい
    - 通信と計算の重ね合わせ + 通信の分割
    - (データによって膜電位計算の刻み幅 = 0.005 ms → 通信時間割合は減少)
  - **データ構造の見直し**
    - キャッシュスラッシングの確実な回避
  - **プロセス並列：ニューロン割り振りの見直し**
    - 実データではニューロン毎の計算量が様々
    - ニューロン数基準のプロセス分割→コンパートメント数・シナプス入力数？

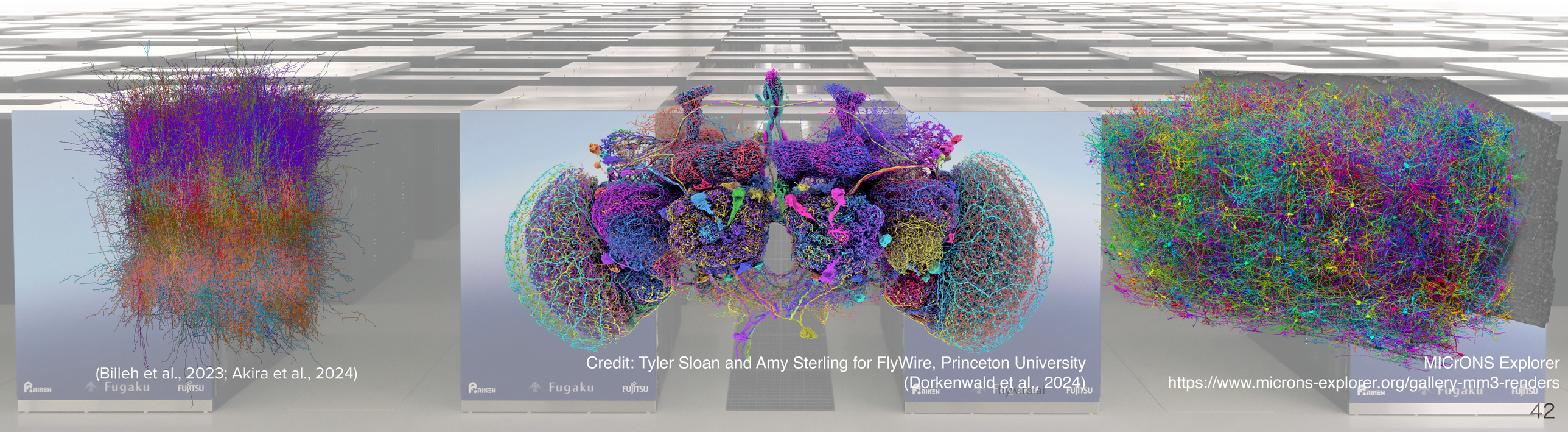
現状の課題と将来の展望

将来の展望

計算機上に脳のコピー「デジタルツイン」を構築

コネクトームデータ+解剖学データからN=1の脳モデルを構築

何が自然発生的に生まれるのか. 学習だけでない様々な脳の情報処理の理解へ



(Billeh et al., 2023; Akira et al., 2024)

Credit: Tyler Sloan and Amy Sterling for FlyWire, Princeton University
(Dorkenwald et al., 2024)

MICrONS Explorer
https://www.microns-explorer.org/gallery-mm3-renders

謝辞

栗山
秋良
井浦
山﨑 (PI)

Gouaillardet
寺澤