

KYUSHU UNIVERSITY · SYS-108

未来の コンピュータ

AI と量子から、計算の仕組みを考える

2026 年 9 月

Olivia Chen Toyama · C2LAB

LECTURE 8



部品から、プログラムが動く仕組みまで

Lec 1	部品と役割	入力・記憶・演算・制御・出力
Lec 2	情報を表す	文字・写真・音をビット列へ
Lec 3	回路で計算する	スイッチからゲート、加算器、ALU
Lec 4	状態を覚える	ラッチからレジスタ、メモリ
Lec 5	命令を実行する	CPU が命令とデータを読み、処理
Lec 6	性能を測り、作る	待ち時間を改善し、回路を設計・製造
Lec 7	OS で共有する	CPU・メモリ・ファイルを安全に管理

ここまでに分かったこと

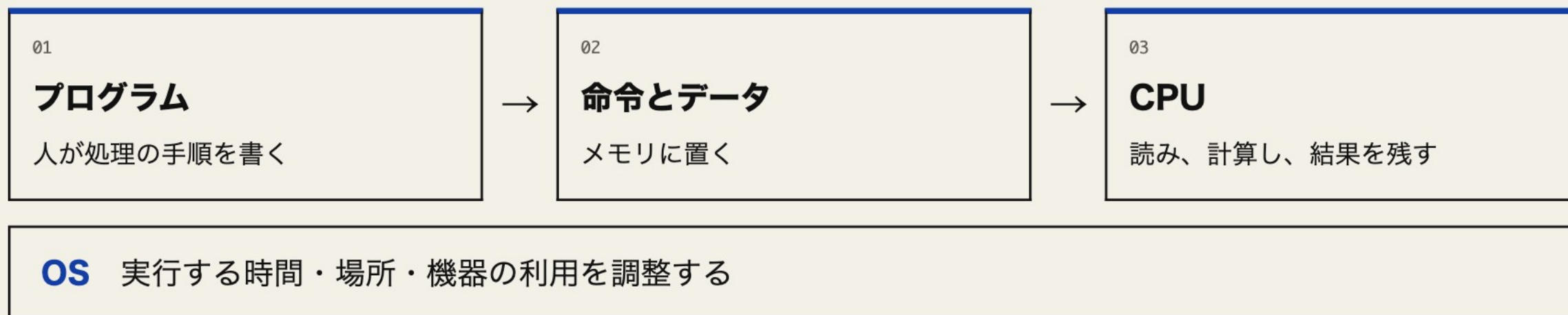
**プログラムが、
機械の上で
動く仕組み。**

今日、開くところ

そのプログラムに、
どんな仕事をさせる？

七つの講義で、情報を表し、計算し、管理する仕組みをつないだ。

動かす仕組みは分かった。何をさせる？



書きやすい手順の例

二つの数を足す。
ファイルを順番にコピーする。

では、この手順は？

写真の中の手書き文字を、
数字として読み取る。

命令を実行できることと、判断の手順を書けることは別の問題。

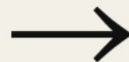
写真に写った「9」を、数字として読む

Lec2 で扱ったもの



写真の中の形

画素の数値として保存できる。



この間の処理を、
どう書く？

集計に使いたいもの



数字としての値

足したり、並べたりできる。

画像の画素を保存することと、何の数字かを判定することは別の処理。

形の規則だけで、全部を判定できる？

提案した規則



輪があって、下に線がある。

例を追加 ①



輪が開いていても、9。

例を追加 ②

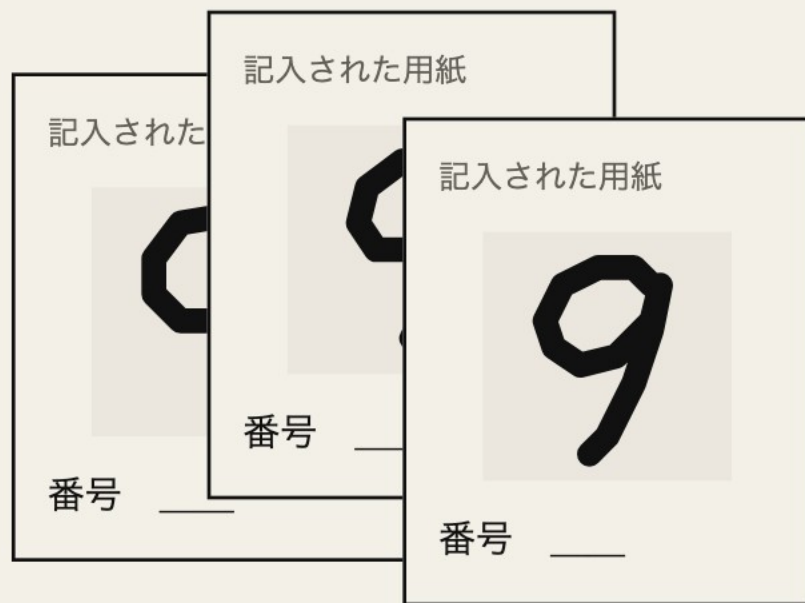


曲線があっても、3。

この規則だけで、筆跡の違いを全部扱える？

手書きの違いを、規則だけで網羅するのは難しい。

読んで入力する仕事を、計算機に任せたい



例えば、たくさんの用紙を集計するとき

**一枚ずつ読んで、
入力する仕事を減らしたい。**

人なら読める。

でも、全ての筆跡に合う規則は書きにくい。

判断の方法を作る、別の手掛かりはないか。

多くの文字を扱うには、筆跡の違いに対応する判断方法が必要になる。

こうした認識や予測を扱う分野が、AI

AI（人工知能） 認識・予測・言語などの課題を、コンピュータで扱う。

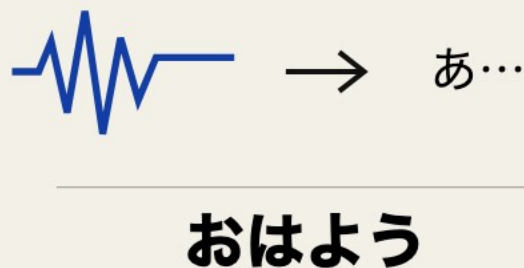
文字を読む



手書きの画像 → 数字の候補

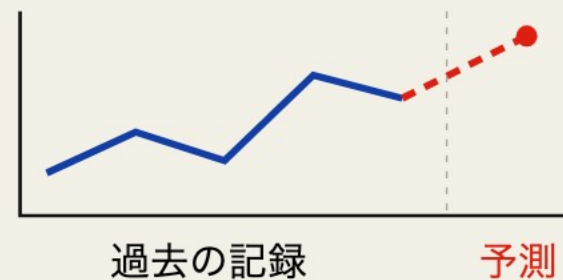
今日、調べる仕組み

音声を文字にする



話した音 → 文字列

需要を予測する



過去のデータ → 次の需要

今日は、AI の一例として「画像から数字を読む」仕組みを調べる。

判断規則を書く方法と、例から学ぶ方法

一つの方法

人が判断規則を書く

輪があり、
下に線があれば……

知識を、条件や手順として表す。

今日、調べる方法

例から判断方法を調整する



多くの入力と正解を使い、
間違いが減るように調整する。

機械学習 データから、判断や予測の方法を調整する。

AI の方法の一つ。ここでは、正解付きの例を使う。

今日見るのは、正解付きの例から判断方法を調整する学習。

花見に行くか。理由の数だけで決められる？

判断の仕組みを、人の例で考える

**天気は悪い。
友だちは行く。
会場は近い。**

三つの条件を、同じ一票として数えてよいだろうか。

+

行く理由は二つ

友だち・近さ

—

行かない理由は一つ

悪天候

理由の「数」と「重要さ」は別。

まず、人の判断で「条件の効き方」を考える。

重要な条件ほど、大きく効かせる

判断の仕組みを、人の例で考える

天気は悪い。
友だちは行く。
会場は近い。

天気を重視する人なら、ほかの二条件がよくても見送るかもしれない。

条件	入力	重要さ	寄与
天気がよい	-1	× 6	-6
友だちも行く	+1	× 3	+3
アクセスがよい	+1	× 2	+2

$$-6 + 3 + 2 = -1 \text{ 見送る}$$

条件がよい：+1、悪い：-1。合計が 0 より大きければ行く、という説明用の基準。

入りに掛ける数を変えると、その条件の寄与が変わる。

同じ入力でも、重みが違えば判断が変わる

判断の仕組みを、人の例で考える

天気は悪い。
友だちは行く。
会場は近い。

同じ条件でも、大切にすることが
違えば判断は変わる。

天気を重視： $6 \cdot 3 \cdot 2$

$$-6 + 3 + 2 = -1$$

見送る

友だちを重視： $3 \cdot 6 \cdot 2$

$$-3 + 6 + 2 = +5$$

行く

掛ける数が、各入力の効き方を変える。この数を**重み**と呼ぶ。ここでは人が決めた値。

重みは、入力が判断にどれだけ影響するかを調整する数値。

では、文字を読む重みは誰が決める？

花見の説明では

人が重みを決めた。

6 3 2

「天気を重視する」という考えを、
数値として書き込んだ。

文字を見分けるときは

よい重みが、まだ分からない。

? ? ?

画像と正解を用意し、
間違いが減るように調整する。

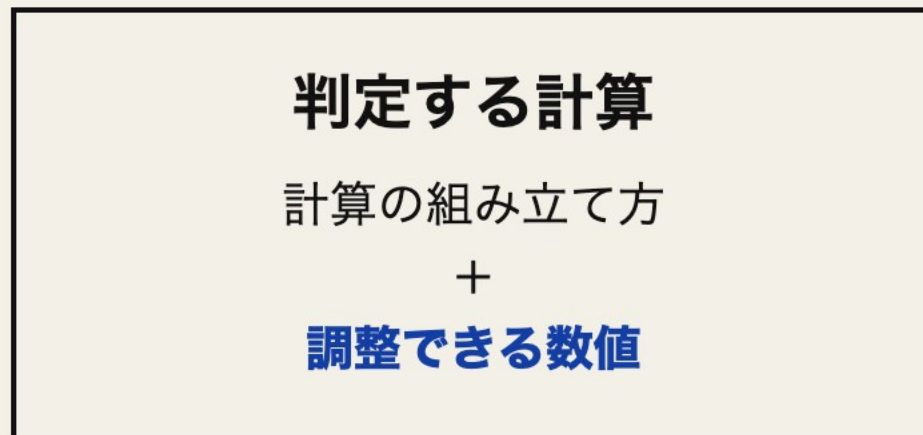
機械学習では、例に合うように重みなどの数値を探す。花見の好みを学ぶ実験ではない。

例と正解を使い、よい重みを探すのが学習。

学ぶとき、何が変わる？



入力：手書き画像



出力：数字の候補

例と正解を使って、中の数値を調整する。

この入力と出力を結ぶ仕組みを、**モデル**と呼ぶ。

ここでは、数値を調整して学ぶ方法を見る。計算方法や学習の手順は人が設計する。

このモデルでは、計算に使う数値を変えて、入力と出力の関係を調整する。

画像を数値にする方法は、Lec2 で見た



28 × 28

784 個の画素値

0.0	0.2	0.8	1.0	0.6	0.0
-----	-----	-----	-----	-----	-----

暗い画素から明るい画素まで、
数値として入力する。

この実演では、28 × 28 の画素を入力する

画素に分けて見る

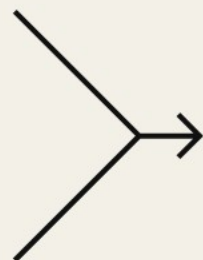
この例では、784個の画素値を入力する。

重みを掛けて、足す。AIも数値の計算

入力 × 重み

$$0.8 \times 2 = 1.6$$

$$0.2 \times (-1) = -0.2$$



二つの結果を足す

1.4

次の計算の材料になる。

この小さな計算を、何度もつなぐ。

値を変換する処理なども組み合わせ、予測を作る。

一部分を取り出した説明例。この和だけで、数字の種類が決まるわけではない。

AIの中でも、入力と重みを使う小さな計算を繰り返す。

学習：予測し、正解と比べ、重みを直す

同じ画像を入力



何の数字だろう？

学習前のモデル

重みを使って計算

重みの一部

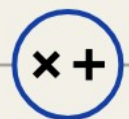
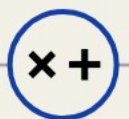
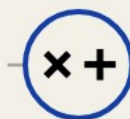
-0.009

重みの一部

0.575

重みの一部

0.348



まず、今の重みで予測する。

モデルの予測

3



3のほうが出やすい予測。

学習の例：3と9を区別する小さなモデル。4枚の例で計算した結果であり、未知の画像への正確さではない。

正解と比べる

最初から

1 / 3 予測する

学習は、例と正解を使って、計算に使う数値を調整すること。

推論：学んだ重みで、画像を層へ通す

学習済みの LeNet-5 系 CNN 重みを固定して、1枚を識別

MNIST の画像

9

3

7

画像

32 × 32

→ 反応を見る

28 × 28 · 6枚

→ 小さくする

14 × 14 · 6枚

→ 組み合わせる

10 × 10 · 16枚

→ 小さくする

5 × 5 · 16枚

→ まとめる

120個の数値

→ つなぐ

84個の数値

→ 予測する

0~9の10候補

一枚の手書き画像

32 × 32



「9」の形は、
中でどう変わる？

次の層の計算結果を、
一つずつ見てみよう。

入力の変更で、他の数字とも比べられる。

まず、画素の濃さを数値にする。ここから同じ画像を最後まで追う。

順に再生

次の層へ

最初から

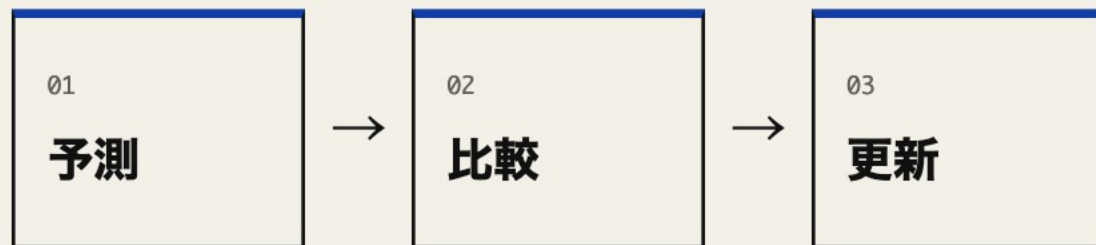
濃い青ほど反応が強い。濃淡は層ごとに調整。表示は実際の計算値。

学習済みの重みを使い、前の層の計算結果を次の層へ渡す。

学習は重みを変え、推論は重みを使う

学習

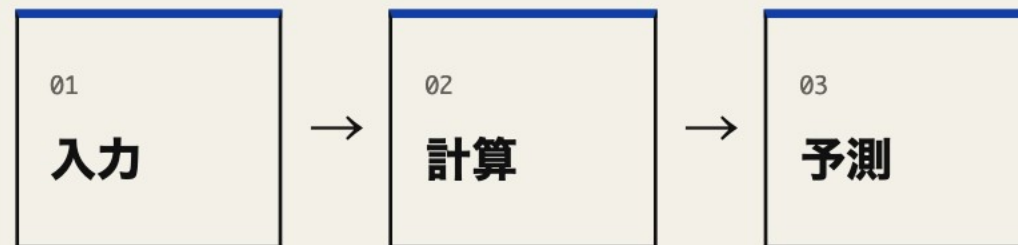
例と正解を使い、数値を変える。



例を変えながら、この過程を繰り返す。

推論

学んだ数値で、新しい入力进行处理。



通常、この一回の予測では重みを変えない。

新しい入力进行处理するたびに、重みを学び直すわけではない。

別の筆跡でも、読めるようになった？

調整に使った文字



9 と予測できた

先ほどの実演で確認したこと。

調整に使っていない文字



未使用の文字で確かめる

学習用の四枚で成功しても、まだ分からない。

実演は更新の仕組みを見るためのもの。性能の評価には、学習に使っていない例を分けて用意する。

学習に使っていない例で、判断できるかを確かめる。

AI の計算は、どんな計算？

学習する仕組みは分かった。

その中で、
何を何回
計算している？

前の層の値 × 重みを、足し合わせる

$$\underline{1} \times \underline{3} + \underline{2} \times \underline{1} = 5$$

$$\underline{1} \times \underline{2} + \underline{2} \times \underline{4} = 10$$

同じ形の「掛ける・足す」が大量に並ぶ。

入力や重みを表に並べると、**行列の計算**としてまとめられる。

二つの入力と二つの出力に縮めた説明例。変換や判断基準の調整値は省略。

大量の積と和を、どう実行するかが次の問い。

同じ仕事でも、組み立て方が違う



CPU

注文に合わせて手順を変える

厨房の比喻は、仕事の組織の違いを表す。料理人の数が、実機のコア数や性能比を表すわけではない。



GPU

同じ形の仕事を多数に分ける

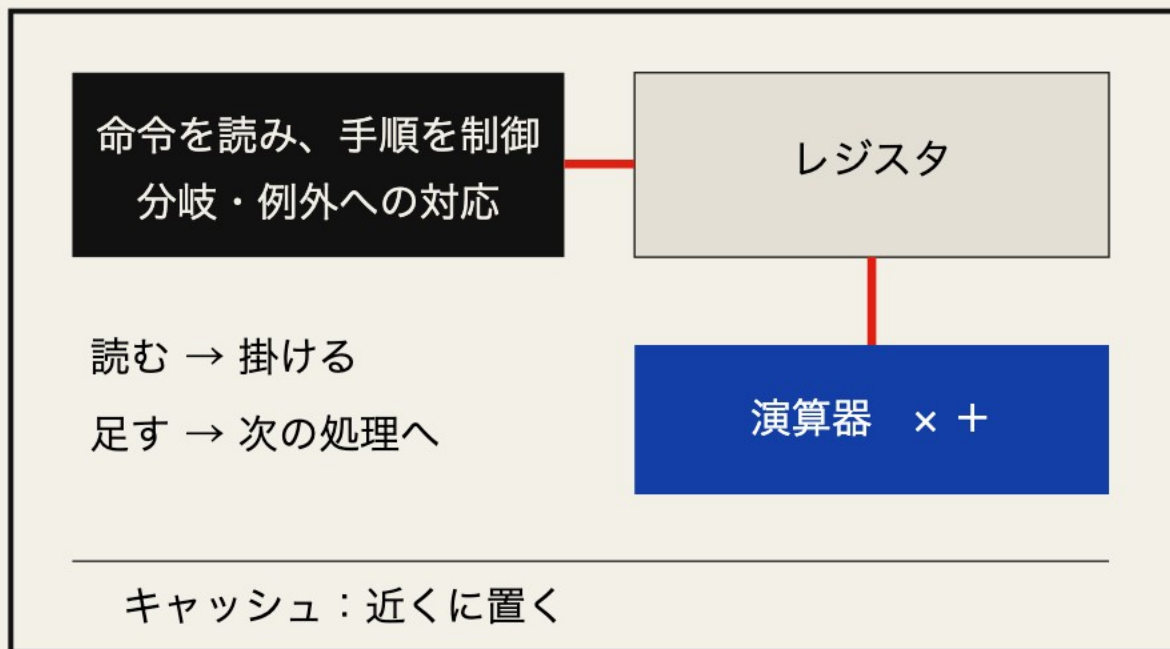


NPU / TPU

よく使う計算と流れに合わせる

CPU・GPU・NPUは、仕事の組織と得意条件が違う。

CPU：いろいろな手順を、柔軟に実行する



CENTRAL PROCESSING UNIT

CPU

柔軟さを支える回路。

多様な命令や分岐、少量の仕事を扱いやすい。

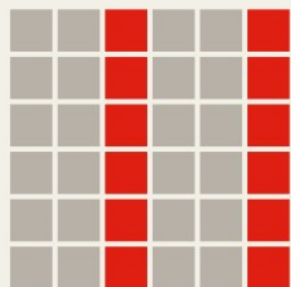
**大量の同型計算では、
制御に使う資源との配分が効く。**

CPUにも複数コア・SIMDがある。AIが計算できないのではなく、仕事の形との相性を見る。

CPUにも並列演算はある。大量の同型計算では資源の配分が効く。

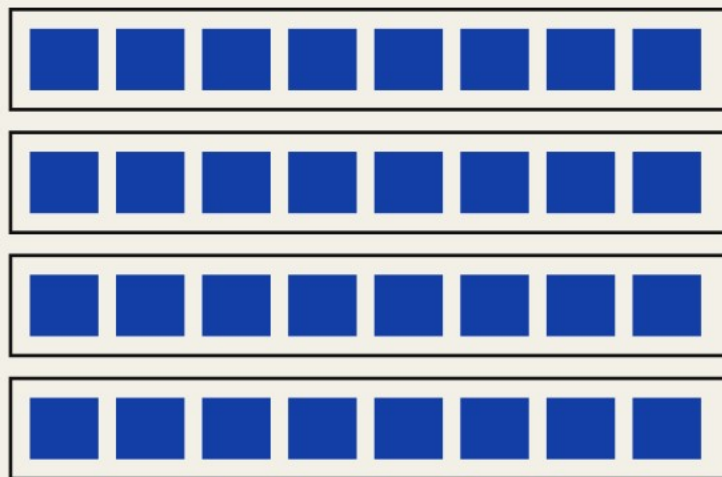
GPU：多数の同じ形の仕事を、並列に進める

出発点：多数の画素を描く



AI の積と和も、
同じ形の仕事が多い。

多数の演算をグループで実行



まとまったデータを供給する

GRAPHICS PROCESSING UNIT

GPU

同じ形の計算を、並列に。

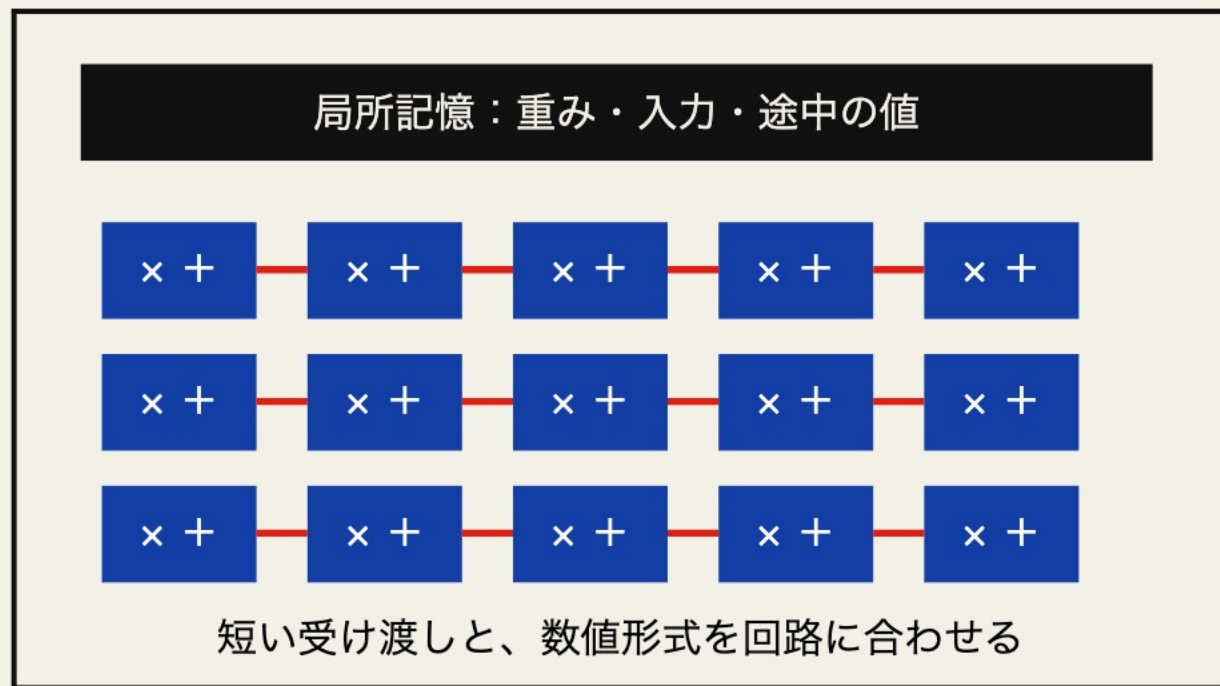
画像処理で発達した多数の演算資源を、
AIの行列計算にも使う。

**仕事をまとめられることと、
データが届くことが条件。**

細かな分岐や小さな仕事は得意とは限らない。現在の
GPUには行列演算専用の回路もある。

GPUは、まとめた計算とデータを供給できるときに強みを生かせる。

NPU：積と和、そしてデータの流に特化する



NEURAL PROCESSING UNIT

NPU

AI の演算に特化する。

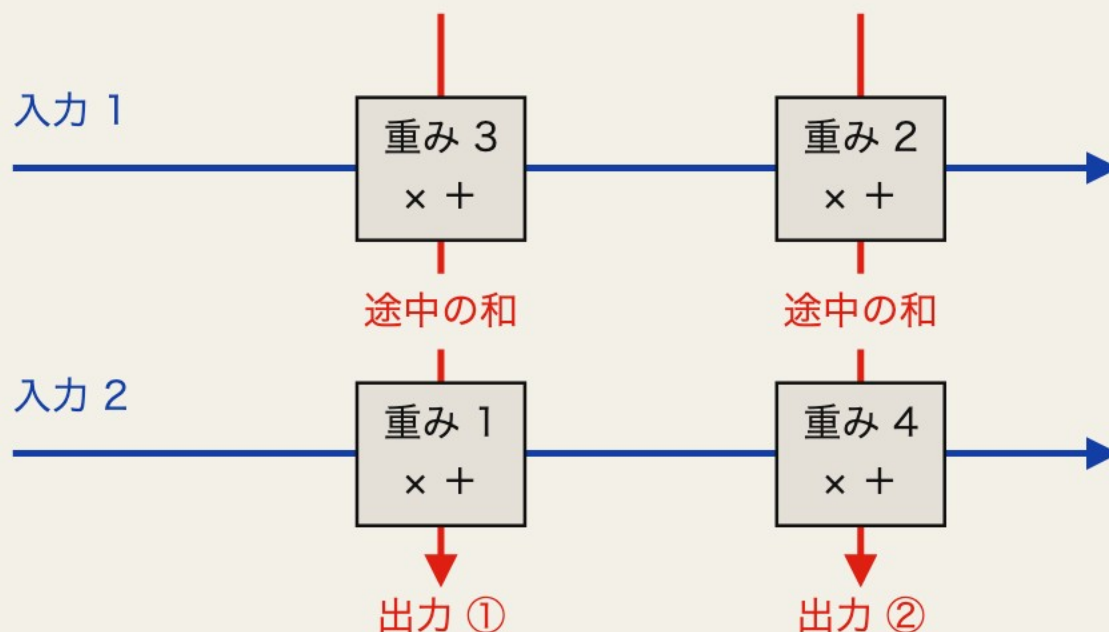
重みを再利用し、必要な精度に絞る。汎用性より電力効率を重視する。

TPU

Google の機械学習用アクセラレータ。
Tensor Processing Unit の略。

演算器だけでなく、記憶と受け渡しも仕事に合わせて設計する。

TPU の考え方：重みを置いたまま、値を流す



行列計算を流れにする

重みを再利用する。

青：入力が横へ進む

赤：途中の和が下へ進む

重みを置く。まず、左の列を上から追う。

流れを再生

次へ

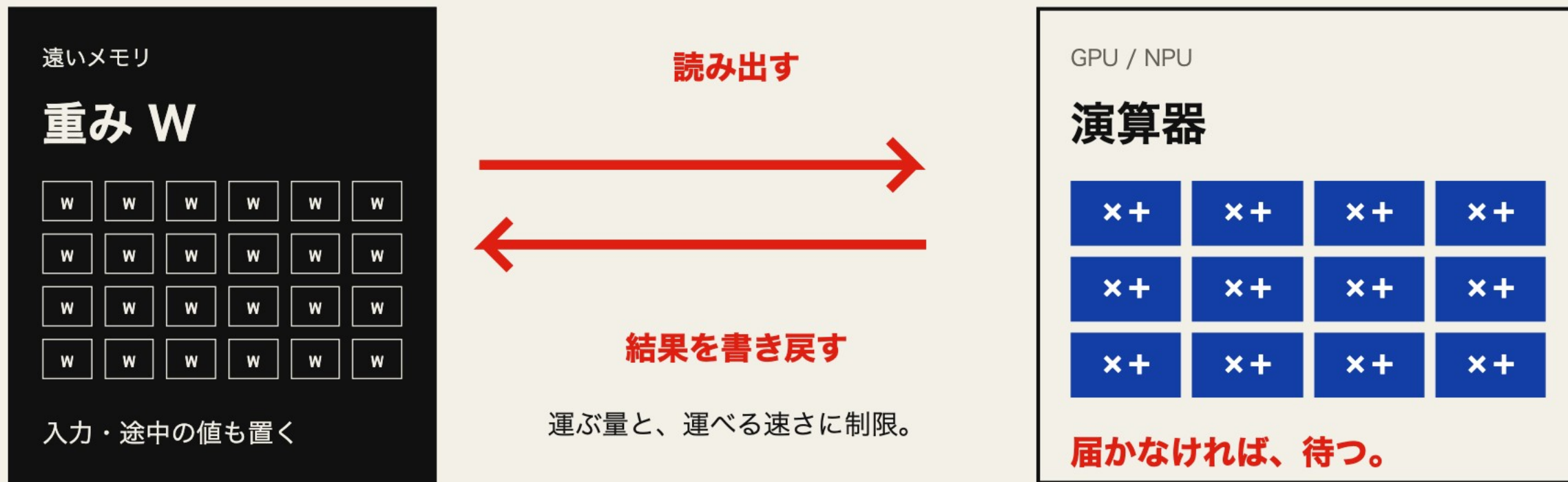
最初から

結果を見る

シストリックアレイを2×2に縮めた図。段階は注目する順であり、実機のクロック時刻ではない。

重みを近くに置いたまま、入力と途中の和を流す。

演算器を増やすと、何が次の制約になる？



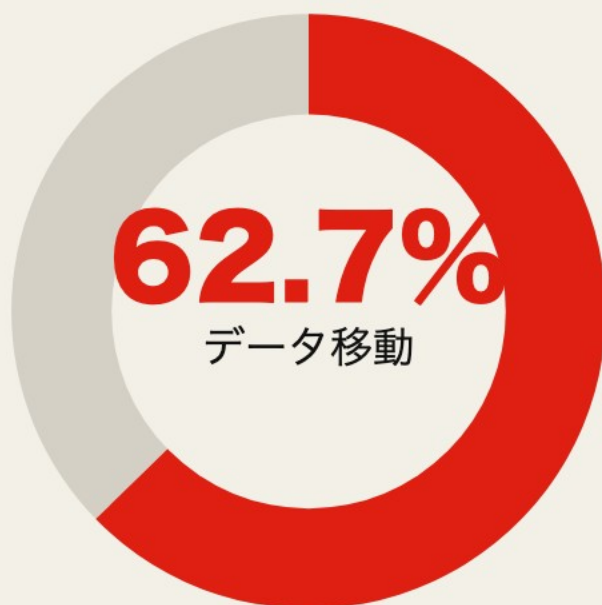
GPUもNPUも、入力や重みが届かなければ待つ。

エネルギーの問題：運ぶことにも費用がある

BOROUMAND ET AL. / ASPLOS 2018

計算するために、
運ぶ電力も使っている。

調査した消費者向け処理では、平均62.7%のシステムエネルギーがデータ移動に使われた。



TensorFlow Mobile の推論：57.3%

Chrome・機械学習・動画処理を含む当時の研究。
現在の全AIや全GPUに共通する固定比率ではない。
灰色37.3%は、演算だけでなくその他の消費を含む。

データ移動は、時間とエネルギーの両方の制約になる。

取りに行く回数と距離を、減らせないか

往復が長い



近くに置いて再利用する



持ってきた値を、何度も使う。

キャッシュ・局所記憶・まとめ処理。
演算器をメモリの近くへ置く方法もある。

近存計算は、演算器をデータの近くへ寄せる考え方。次は、メモリの中で計算する場合を見る。

近くに置いて再利用することが、移動を減らす第一歩。

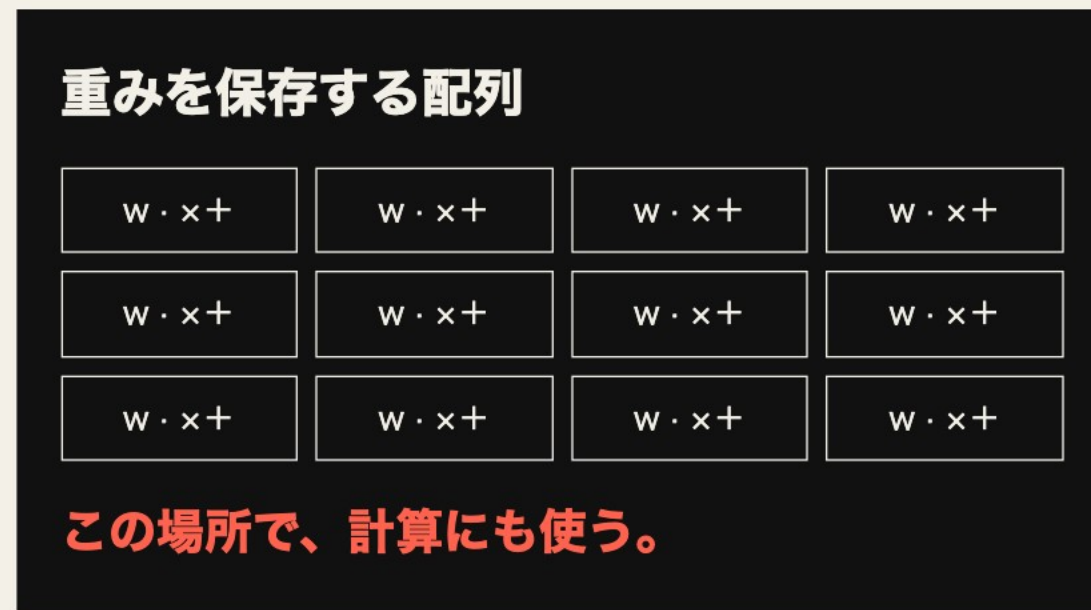
もう一步：記憶している場所で、計算する

従来：記憶と計算を分ける



重みを読み出して、演算器へ運ぶ。

IN-MEMORY COMPUTING



計算を、データのある場所へ。

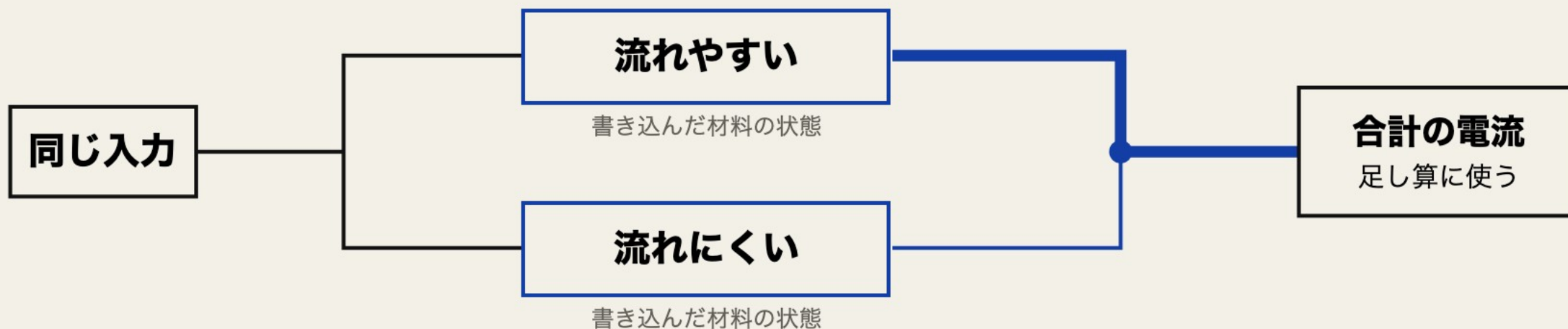
In-memory computing は、データのある場所で一部の計算を行う。

流れやすさに重みを覚え、電流を集める

入力を電圧にする

流れやすさに、重みを記憶

電流を集める



RRAMという素子を使う一例 記憶した重みを、離れた演算器まで運ぶ手間を減らす。

線の太さは電流の違いを表す。実際には、扱える精度や数値への変換も考える。

材料の性質を、記憶と計算の両方に利用する工夫がある。

移動を減らす工夫にも、確かめる条件がある

減らしたいもの

データを運ぶ手間



移動の回数と距離が減れば、
時間や電力量を減らせる可能性。

一緒に確かめるもの

結果の精度と、周りの処理

精度

材料のばらつきや雑音で、
結果がずれないか。

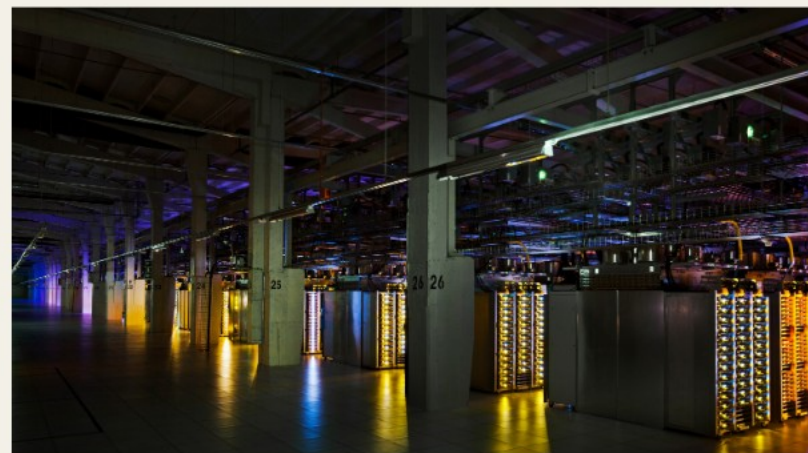
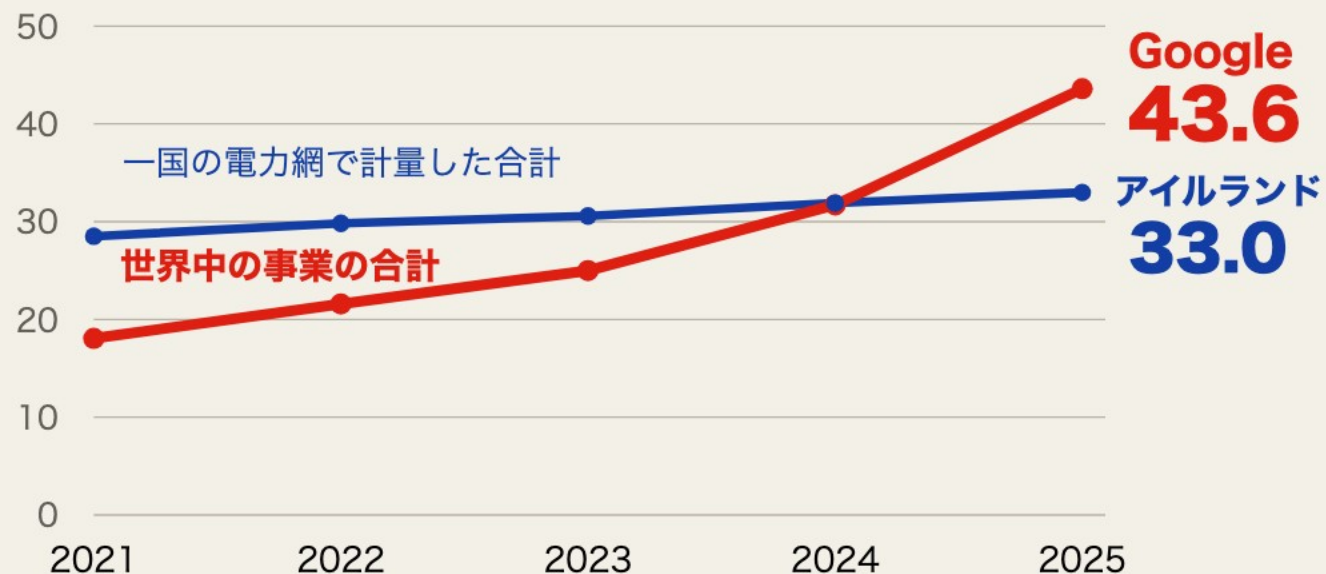
変換

数値と電圧・電流を変換する
処理にも、時間と電気を使う。

一つの部品の利点を、計算機全体の効果として確かめる。

一社の電力量が、一国を上回る規模に

年間の使用電力量 (TWh)



画面の向こうには、これだけの設備。

Google / Hamina, Finland · 写真：Google

2025年、Googleの電力量は
アイルランドの計量値の約 **1.32倍**。

Google 2026環境報告 p.93 (過年度再計算値) / CSO 2026年7月。GoogleはAI以外・オフィス等も含む世界合計。国との比較は規模の目安。1 TWh=10億kWh。

計算の需要は、発電・送電の規模にまで影響する。

電気はどこから来る？ 熱はどこへ逃がす？



Google / Hamina, Finland · 写真：Google



Crane Clean Energy Center · 既存施設写真：Constellation

熱を逃がす 冷たい海水を利用。

海に近い立地と、熱交換器・冷却配管。

寒い場所でも、冷却設備は必要。

電気を確保する 発電所単位の契約。

Microsoftが**20年間**の電力購入契約。

835 MWの原子炉の再稼働を支える。

資料：Google/Constellation（2024年契約）。835 MW＝発電能力。Constellation所有・電力網経由の供給。写真は再稼働前。

チップだけでなく、冷却・電源・立地も計算機的设计になる。

一回の要約で、どれくらい電気を使う？

論文を読み込ませて、要約を作る算例

「この論文を要約して」



約2.5 Wh

一回の処理で使う電力量

同じ電力量を、身近な照明で考える

10 W の LED 照明

≈



15 分

$10 \text{ W} \times \frac{1}{4} \text{ 時間} = 2.5 \text{ Wh}$

入力 約10,000 → 出力 約500 トークン

同じ仕事でも、モデルや処理量で変わる。

Epoch AI (2025)、GPT-4oを想定した推計。現在のChatGPTの実測値ではない。トークン=文章を分けた処理単位。

一回の処理にも電力量があり、その量は条件によって変わる。

AIを支えるのも、計算・記憶・電源

画像も、重みも

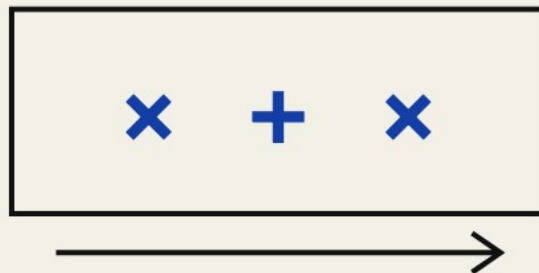


0.8
0.2
...

数値として覚える

数値をメモリに置き、
計算で使える形にする。

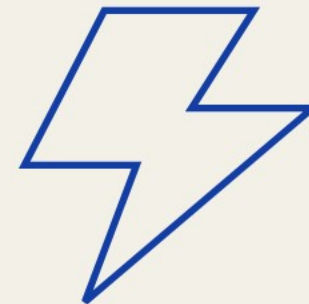
予測のために



計算して、運ぶ

演算器を働かせるには、
入力や重みを届ける。

装置を動かすために



電気を使い、熱を出す

規模が大きくなると、
電源と冷却も大きな課題。

AIの規模が大きくなるほど、計算とデータ移動を支える設計が大切になる。

別の問い：自然のふるまいを、どう計算する？

同じ炭素でも、違う性質



グラファイト（黒鉛）



ダイヤモンド

原子や電子のふるまいから

**物質の性質を
計算で調べたい。**

小さな世界の量子の性質を、
計算する側にも使う。

それが、量子計算を研究する
出発点の一つ。

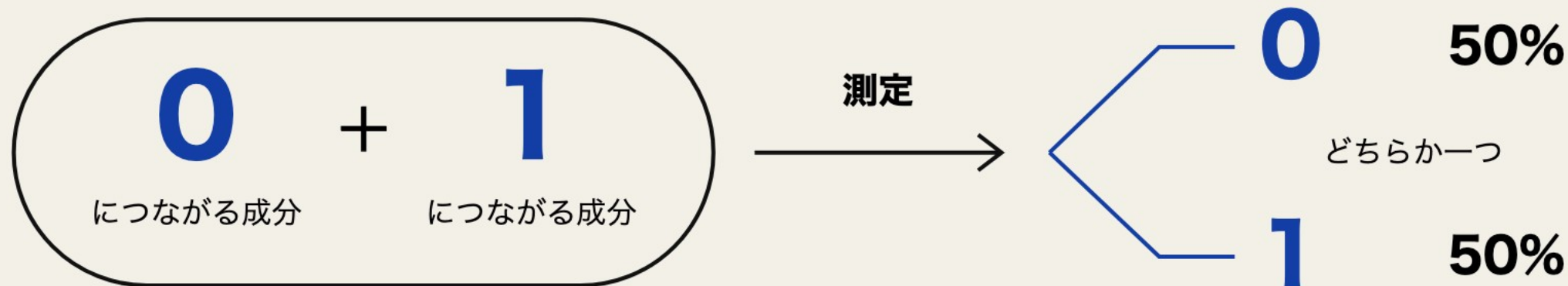
自然の量子の性質を、計算するための道具として使う。

量子ビット：二つの成分を、一つの状態に

ビットは、0 または 1。

量子ビットは、二つの成分を重ね合わせた状態も使う。

1個の量子ビット

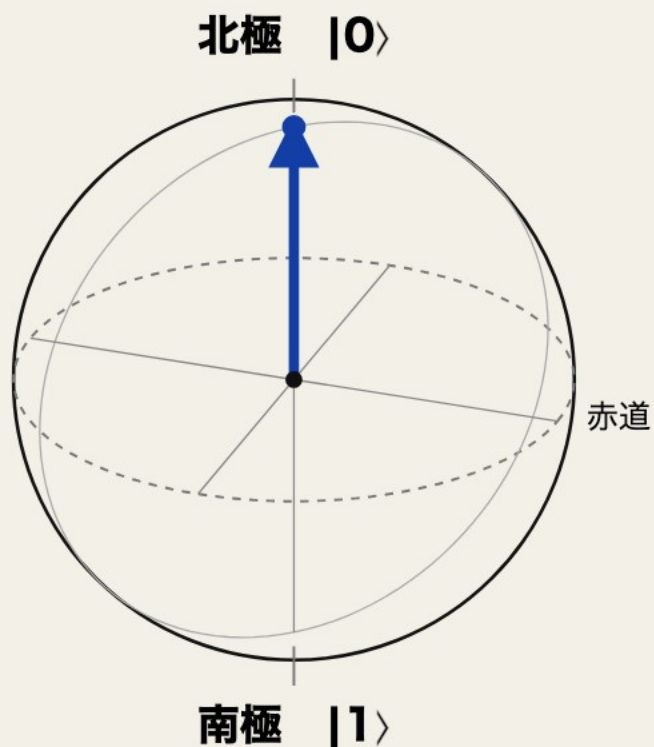


図は、0 と 1 が同じ確率で出る重ね合わせの例。

毎回同じ状態を用意して測ると、長い目では半分ずつになる。

重ね合わせも、一つの量子ビットの状態。測定で読む結果は一つ。

量子ビットの状態を、ブロッホ球で見る



一つの量子ビットの状態を表す地図

矢印の向きが、状態を表す。

北極：0 の状態 赤道：重ね合わせ 南極：1 の状態

測定では、**0 か 1 のどちらか**を読む。

赤道へ動かす

最初から

今の状態を、0 / 1 で測ると

0 100%

1 0%

赤道上の向きの違い（位相）も、操作後の結果に効く。

球は電子の軌道ではない。球面は単一量子ビットの純粋状態を表す。

矢印の向きを操作し、測定で 0 / 1 の結果を読む。

測る前の操作で、結果の出やすさを変える

途中では測らず、操作を重ねる

0が出る状態を用意



まず、測定で0が必ず出る状態から。

実際の問題では、目的に合う
操作の手順を作る必要がある。

状態の操作を示す小さな例。計算の高速化を示すものではない。

いま測定すると（理想的な割合）



同じ準備と操作を最初から繰り返し、
測定結果の出やすさを確かめる。

操作を1回行う

最初から

1 / 3

測る前の操作が、結果の出やすさを変える。

量子計算では、問題に合うように状態を操作し、測定結果を利用する。

どんな問題で、利点が期待される？

物質の性質を調べる



グラファイト（黒鉛）



ダイヤモンド

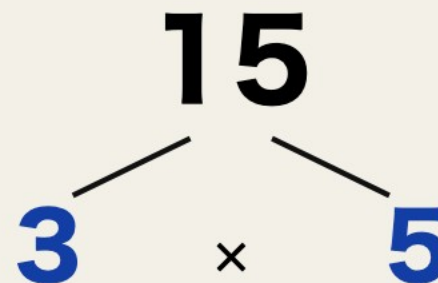
原子と電子のふるまいから、
物質の性質を予測したい。

対象の系・精度・古典手法との比較が必要。

図は課題の意味を示す例。小さな「15」の分解は、量子計算の優位を示す実験ではない。

目的に合う手順と、十分な精度の装置がそろって初めて役立つ。

整数を因数に分ける



大きな整数を分解する手順は、
暗号の安全性にも関わる。

多数の量子ビットを誤りから守る装置が必要。



量子を支える装置 / 超伝導回路の例

量子チップを、 極低温で動かす

動作温度の一例

0.015 K

絶対零度より、わずか 0.015 度高い。

冷やす

熱による乱れを抑え、
量子状態を使えるようにする。

つなぐ

外の装置から操作の信号を送り、
測定結果を読み出す。

計算するチップを、
冷却・配線・制御が支える。

IBM / 希釈冷凍機の内部
Photo: Graham Carlow, 2018

量子ビットを作る、三つの方法



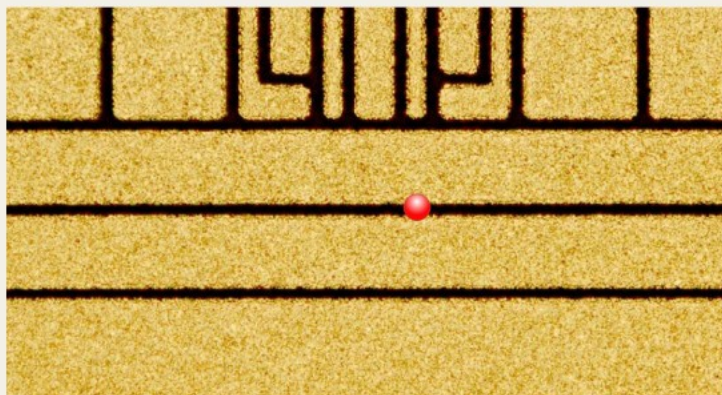
IBM Research / 冷却装置

超伝導回路

人工回路の状態

極低温で保ち、
マイクロ波で操作する。

装置全体、顕微鏡像、原子配列の画像を並べた比較。尺度は異なる。原子数だけで計算性能は決まらない。

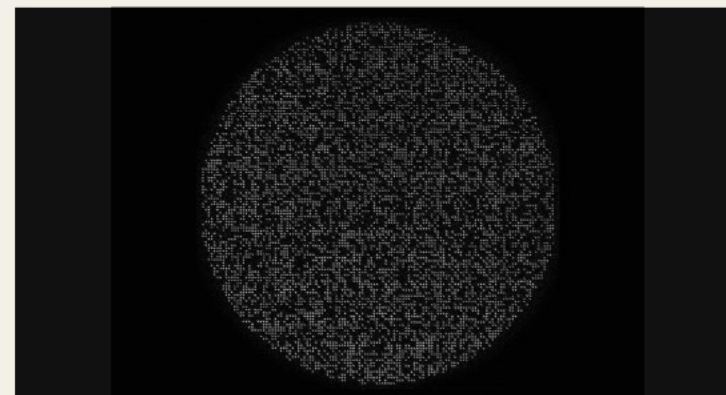


NIST / 着色顕微鏡像

トラップドイオン

電荷をもつ原子

電磁場で閉じ込め、
光などで操作する。



Caltech / 原子配列の画像

中性原子

電荷が中性の原子

光のピンセットで並べ、
レーザーで操作する。

回路・イオン・中性原子。それぞれに制御と誤差の課題がある。

数を増やすと、状態を守ることも難しくなる

1個ごとの、わずかなずれ



10個を守る 1マス=1個・1区画=100個

独立した小さなずれの説明モデル。実機の予測値ではない。一致度は、計算の正答率とは異なる。

全体の状態との一致度は下がる



再生

次へ

最初から

1 / 3

一つずつの小さなずれが、全体に積み重なる。

量子ビットの数とともに、精度と保護も高める必要がある。

量子計算も、古典計算に支えられる

計算全体を支える

古典計算機

操作を指示する

測定結果を整理する

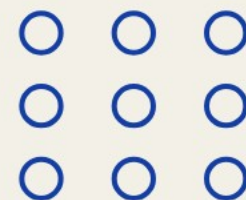
結果を確かめる

操作の指示 →

← 測定の結果

量子状態を使う

量子プロセッサ

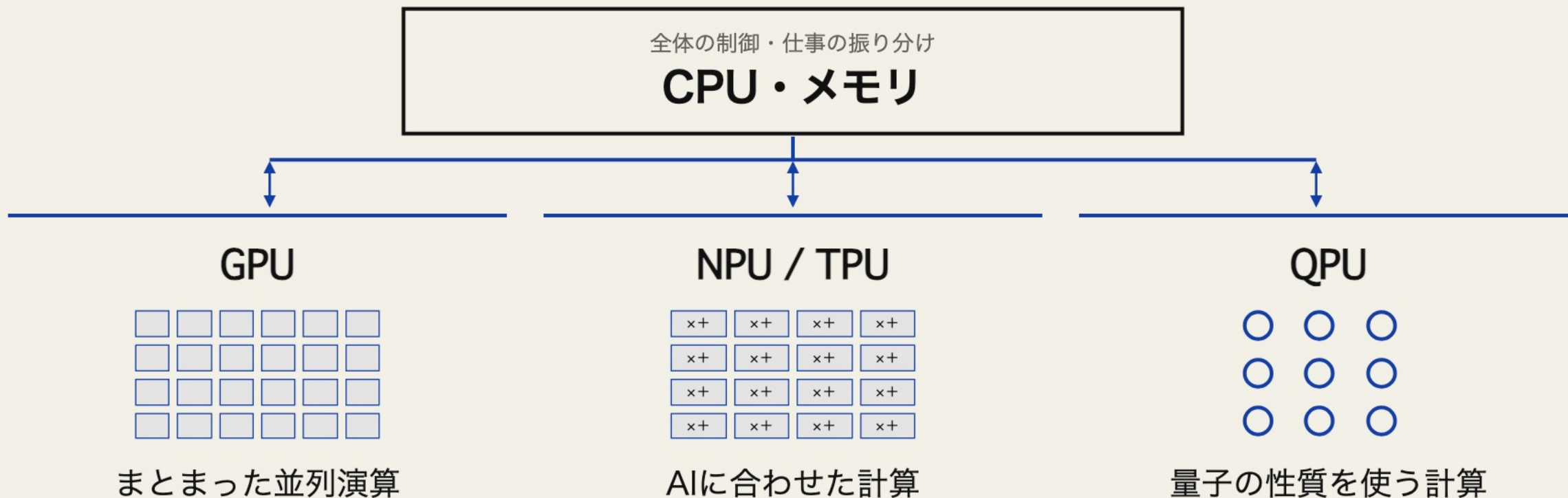


準備 → 操作 → 測定

量子の計算を成立させるために、**古典の制御と処理**が必要。

量子の状態操作と、古典の制御・検証を一つの系にする。

将来の計算機は、得意分野を組み合わせる



将来の構成の一例。各方式の得意分野と、データの受け渡し・制御の費用を併せて評価する。

一つの方式に統一せず、適した仕事を適した計算機へ。

今日見た工夫を、これまでの知識につなぐ

今日見た工夫を、学んだ層へ戻す。

Lec 2 表す 数値・ビット	Lec 3 計算する 論理回路	Lec 4 覚える 状態・記憶	Lec 5 実行する CPU・メモリ	Lec 6 改善する 性能・設計	Lec 7 管理する OS・資源
------------------------------	------------------------------	------------------------------	---------------------------------	-------------------------------	-------------------------------

Lec 1 全体を、一つのシステムとして見る

重みと数値

情報をどう表すか

データの再利用

記憶をどこに置くか

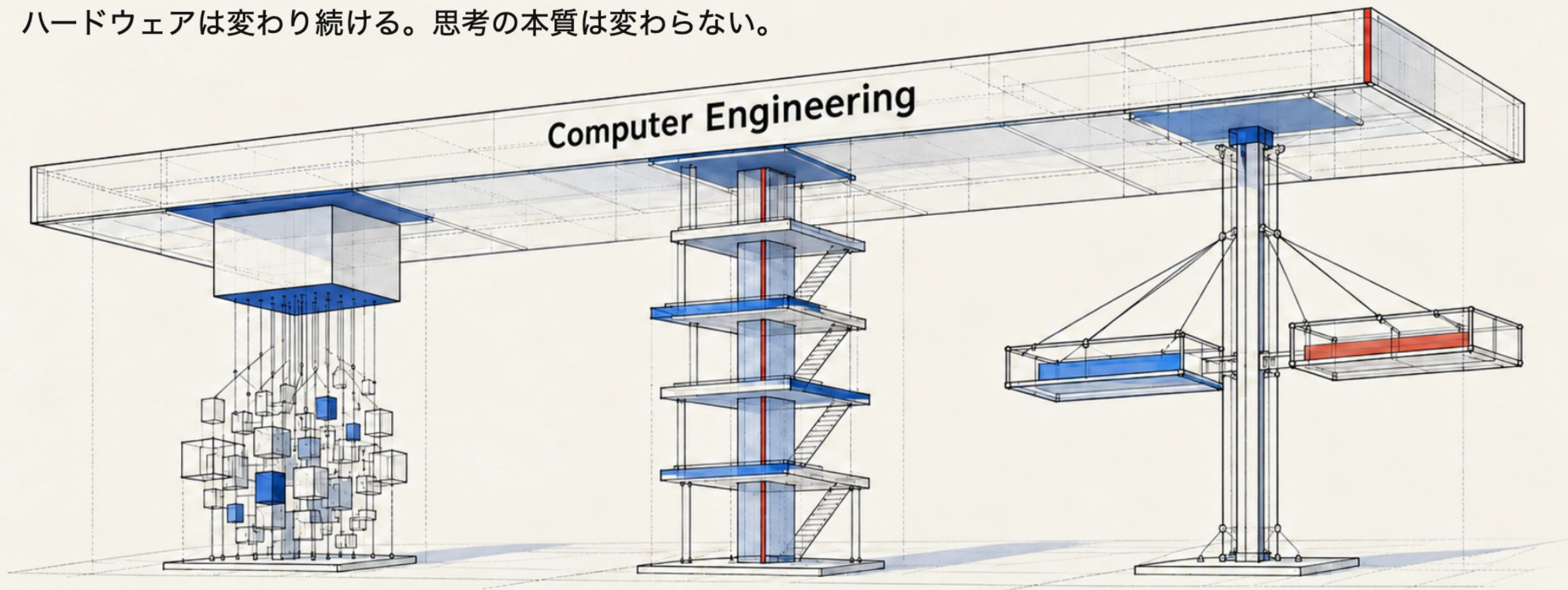
制御と役割分担

装置を動かし、結果を確かめる

細部を分け、つなぎ方と取捨選択を考える。

変わる技術、変わらない思考

ハードウェアは変わり続ける。思考の本質は変わらない。



抽象化

Abstraction

細部を隠し、本質を捉える

階層化

Hierarchy

複雑さを分け、積み上げる

トレードオフ

Trade-off

目的に合わせて、選ぶ取る