Virtual
AI
Explores

AMI System
非同期処理
基礎知識

2024/02/14 GesonAnko

自己紹介

- げそん (GesonAnko)

  𝕏 (Twitter)@GesonAnkoVR

- ML集会 主催

  自律機械知能の研究開発

  PythonでML関係ツールの作成

  VRChatに P-AMI<Q> っていう自律機械知能を作ったよ。

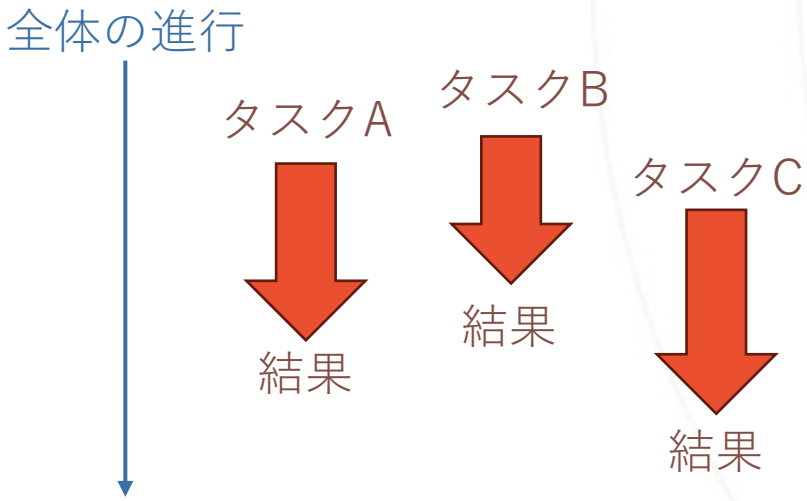
目次

非同期処理とは

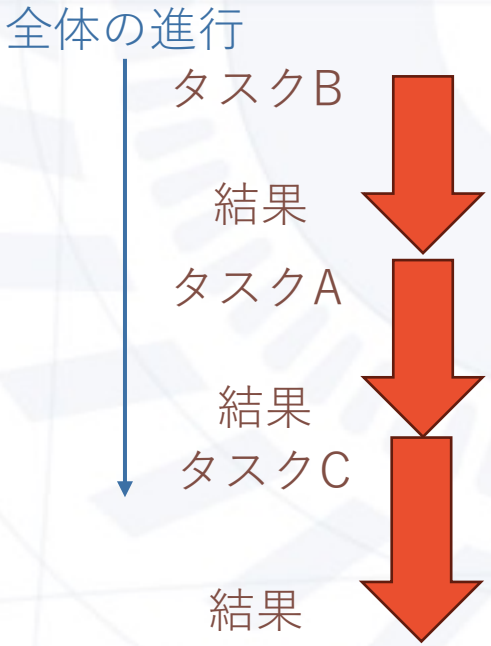
- あるタスクを実行中に、別なタスクを実行する。
  - Ex1. UIは動いていて欲しいけど、裏では実処理を回したい。
  - Ex2. サーバーからのレスポンスを待っている間、別の処理をしたい。

※スレッド ≒ タスク。同時実行されるタスクのこと

非同期処理の場合



同期処理の場合



非同期処理のメリット・デメリット

Virtual AI Explores

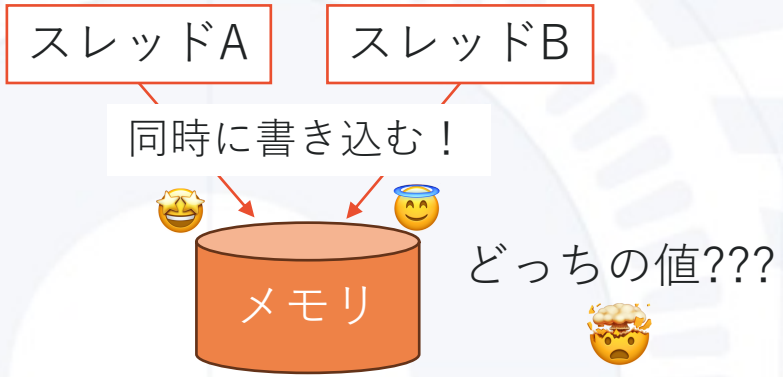
- メリット
  - 上手く処理を組めれば、マシンパワーをフル活用できる
  - UIなどのずっと動いていて欲しい処理を作れる

- デメリット
  - プログラムが複雑になる。全体が見通しにくい
  - デバッグが難しい。（タイミングなどが重要になったり…）
  - メモリ安全性問題が発生する（後述）

メモリ安全性と排他制御

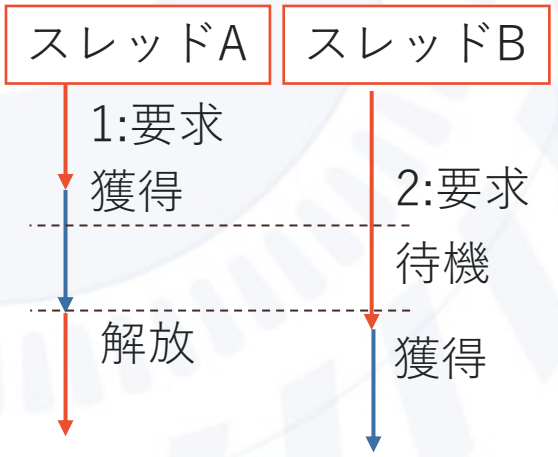
- 安全性問題

  同じメモリに複数のスレッドから同時に書き込めてしまう。
  - どっちの値になるかわからない
  - 壊れるかもしれない。
  - メモリを読み取り中に消されるかもしれない



- 排他制御

  同じメモリ領域への同時アクセスを制限する。
  - **Lock**: スレッドが一度ロックを獲得すると、それ以後のロック獲得の試みはロックが解放されるまでブロック。
    - 実際に使うのは RLockだけどね



  - 他にも “Condition”, ”Event”, “Barrier”, “Semaphore” などがある。

書き込み可能権限と読み取り専用オブジェクト

Virtual AI Explores

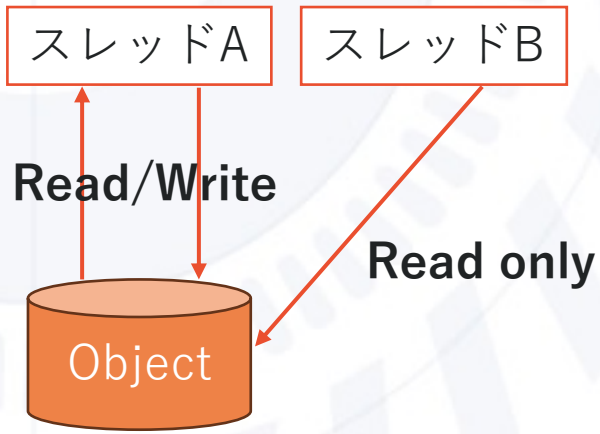
- **書き込み可能権**

全てのオブジェクトは**たかだか1つのスレッドが書き込み可能である。**
スレッド間で共有されたオブジェクトに対して考える。

書き込み可能権限

　排他制御をしても、複数のスレッドから同じオブジェクトを変更する処理は、
オブジェクトの状態を追えなくなるので危険。

- オブジェクトはただ一つのスレッドから書き込みされる。
- 権限のないスレッドは、読み取りのみ可能である。



オブジェクトは書き込み可能なスレッドが生成し、
読み取り専用オブジェクトとして他のスレッドへ共有することが多い
ただ、権限は移動できる。(Queueや Pipeなどで)

非同期更新 AMI System

Pythonで非同期処理を行うために

Virtual AI Explores

- “**threading**” モジュール

  ”target”に関数を指定、”args”を引数に与えて、 “start”メソッドで開始

  メモリはスレッド間で共有される

  → 基本どんなオブジェクトも共有可能

```
from threading import Thread
t = Thread(target=print, args=[1])
t.start() # Start in background.
# Console output: 1
t.join()
```

※ Global Interpreter Lock (GIL) で本当に同時に動くスレッドは一つのみ…

→ 性能向上のために使えない。（ただしI/O系は良い）

- 他にも

  ”multiprocessing”, “asyncio” などがある。

  ”concurrent.futures” モジュールは更に高レベルのAPIを提供

AMI Systemのスレッドの種類・動作方式

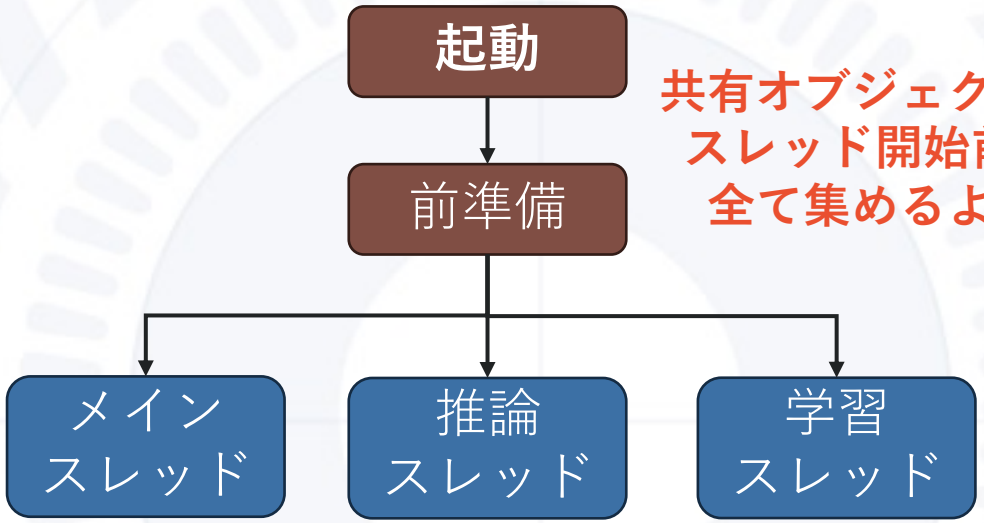
- 種類
  - Main スレッド：システム制御、WebAPI
  - Inference スレッド：VRChatとやり取り
  - Training スレッド：深層モデルの学習

  → 大枠の並列数は静的に**3**とする。

- 動作方式の Pros / Cons

  Pros
  - デバッグが容易。テストを実行しやすい

  Cons
  - スケーラビリティ❌
  - 動的なシステム変更に対応しにくい



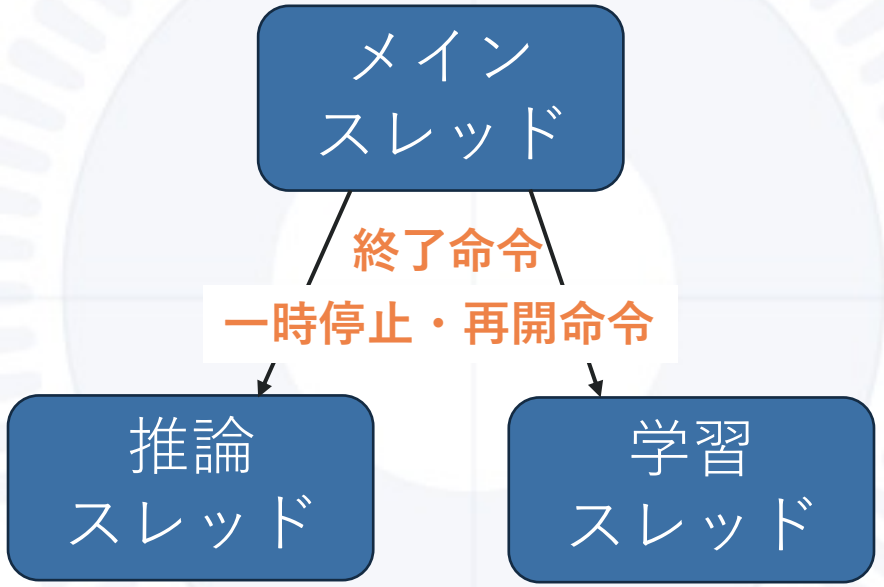
共有オブジェクトは
スレッド開始前に
全て集めるよ！

Virtual AI Explores

共有オブジェクト

システムコントロール

- 所有構造
  - Main スレッド から Inference, Trainingへ共有
- 共有情報
  - システムの終了フラグ（`System.is_active()`）
  - 一時停止＆再開（`System.pause()`）
  - その他
    状態保存命令とか…?

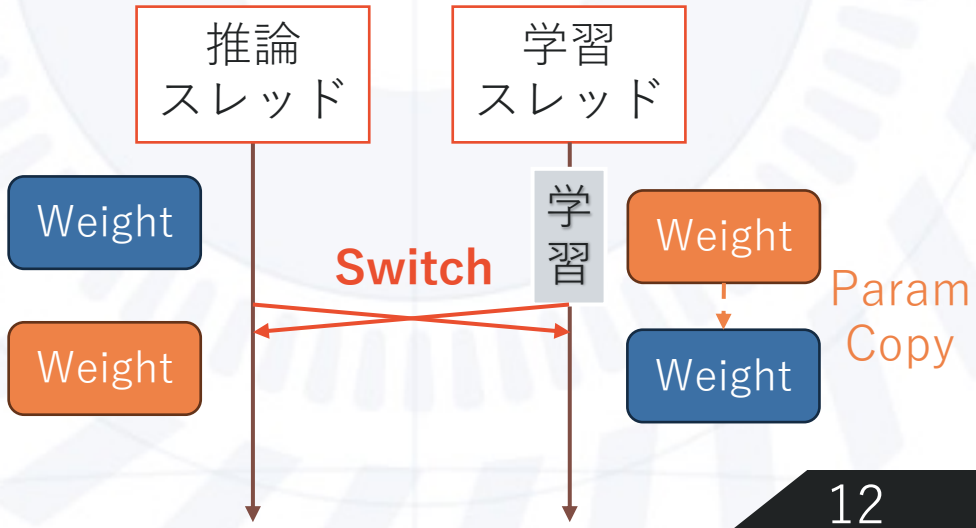


深層モデル

- 書き込み権限構造
  - Trainining スレッド (writable) | Inference スレッド (read only)
- スレッド間共有に際して
  - 学習用モデル (Writable) を推論用モデル (Read only)にして共有
  - 推論用モデルは学習されたモデルが、推論モードで渡される
    → 現在の深層モデルは推論フローと学習フローで挙動が異なる
  - 学習が終わったら推論モデルにパラメータ同期

同期方式

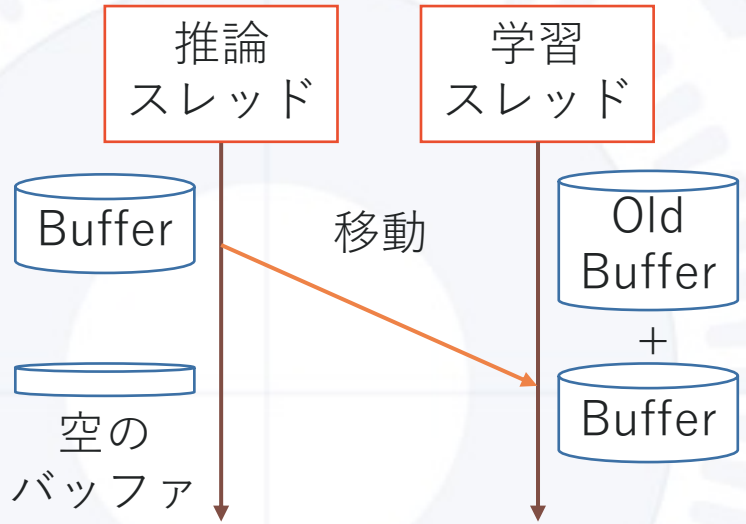
- 参照 Switching
  推論スレッドへの **高速な**同期システム
  Switch後、推論モデルから学習モデルへコピー
  制約：モデルが内部状態を持てない。



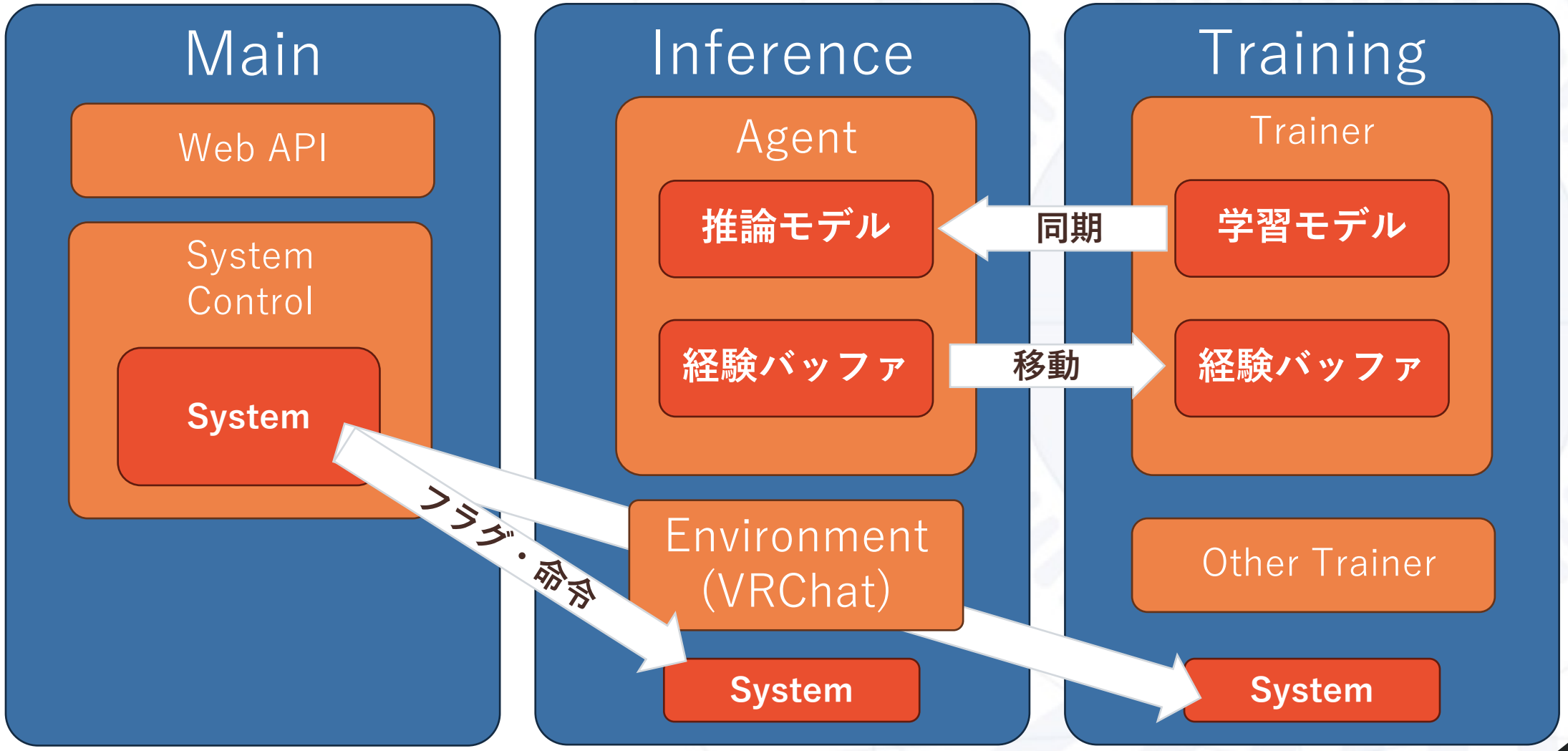
経験バッファ

- オブジェクトの書き込み権限
  - 権限（オブジェクトそのもの）が**移動**する
  - Inferenceスレッド → Training スレッド

- 共有時の挙動
  - Inferenceスレッドはバッファにデータを貯める
  - Trainingスレッドはバッファを奪い取る
    → データポインタごと移動して**空のバッファにリセット**



想定される全体像


Main
Web API
System Control
System
Inference
Agent
推論モデル
経験バッファ
Environment (VRChat)
System
同期
移動
フラグ・命令
Training
Trainer
学習モデル
経験バッファ
Other Trainer
System

参考文献

- Lockオブジェクト
  https://docs.python.org/ja/3.11/library/threading.html#lock-objects
- スレッドベースの並列処理
  https://docs.python.org/ja/3.11/library/threading.html#module-threading
- 所有権とは？（参考：Rustのより厳密な所有権システム）
  https://doc.rust-jp.rs/book-ja/ch04-01-what-is-ownership.html

この資料のリンク

ソースコード

Virtual AI Explores

プロジェクト
メンバー募集中！

募集

- 現状

  既に Myxyさん、ゆんたんさんが参戦

  基礎設計は終盤。現在は実機能の実装が多くなってくる時期。

  → センサ系、アクチュエータ系、既存コードの引き継ぎ…

- 要件
  - PyTorchなどでMLのプログラムを書いた事がある方
  - C++/C#など、静的型付け言語（または mypy）を使用した経験
  - GitなどのCIを用いた開発経験
  - ~~非同期処理を書いた事がある方~~
  - 他人のコードを読める人（作るときに把握すべき領域のみ）