フルスクラッチ LLM「PLaMo™」の事前学習を支える技術

Preferred Networks

Google Cloud Next Tokyo

廣川　祐太

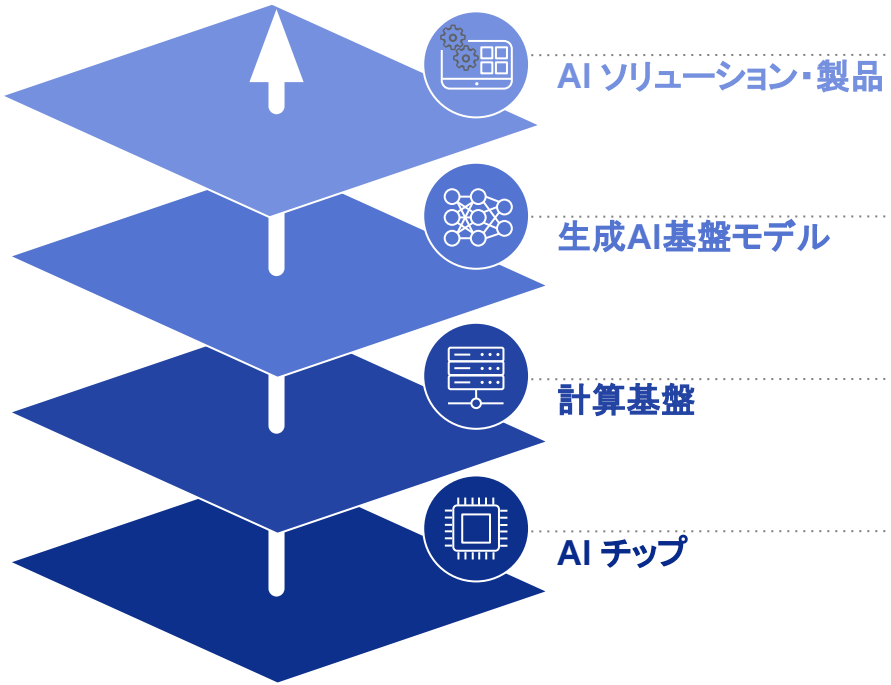
株式会社 Preferred Networks
LLM開発事業本部 LLM開発部
事前学習チーム
エンジニア

目次

01. フルスクラッチ LLM「PLaMo™」

PFN の事業: AI 技術のバリューチェーンを垂直統合

PFN は、チップ、計算基盤、生成 AI 基盤モデル、ソリューション・製品まで、AI 技術のバリューチェーンを垂直統合し、ソフトウェアとハードウェアを高度に融合することで、競争力の高い技術の開発および産業応用を進めています。



様々な産業向けの AI ソリューション・製品

PreferredAI
MiseMise
MATLANTIS
PFN 3D Scan
Preferred Networks Visual Inspection
kachaka

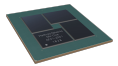
PLaMo
PLaMo Prime（国産 LLM）
PLaMo Lite（エッジ向け SLM）

PFP
物質のエネルギー計算モデル


GPU クラスタ


MN-3
(MN-Core™
クラスタ)

PFCP
MN-Core™ 2 を
計算資源とした
クラウドサービス


MN-Core™

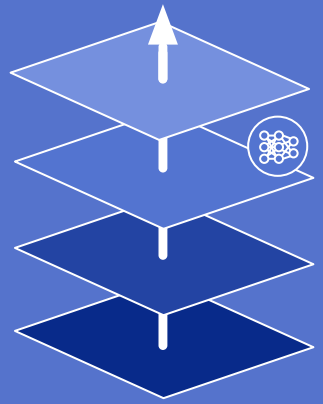

MN-Core™ 2


生成 AI（推論）向け
MN-Core L1000
（2027 年提供予定）

MN-Core
次世代

Preferred Networks

生成AI・
基盤モデル



PLaMo™

PFN グループの生成 AI 基盤モデル

- 標準的な API で利用可能
- 1M トークンあたり 60 円 (入力) / 250 円 (出力) と、gpt-4.1-mini とも競争力のある価格を実現
- コンテキスト長 32 Ki

世界最高クラスの日本語性能

- 主要な日本語ベンチマークにおいて GPT-4o を超える精度
- Nikkei Digital Governance (外部機関) 評価で GPT 等をおさえ基礎言語力 1 位 (2025 年 5 月時点)

純国産フルスクラッチモデル

- ゼロから事前学習を行い、不明瞭な点の存在しないモデル
- 独自開発モデルのため、高度なカスタマイズが可能

API で簡単に導入可能

- 国内サーバーに実装したモデルでの API 提供
- AWS Bedrock MarketPlace を介した独立サーバーでの推論能力提供が可能

柔軟な導入形態

- チャット形式に加え、API やオンプレミス環境でも利用可能

幅広いモデルサイズ

- 大規模モデルに加え、エッジデバイスで動作可能な軽量モデルも提供

専門家によるサポート

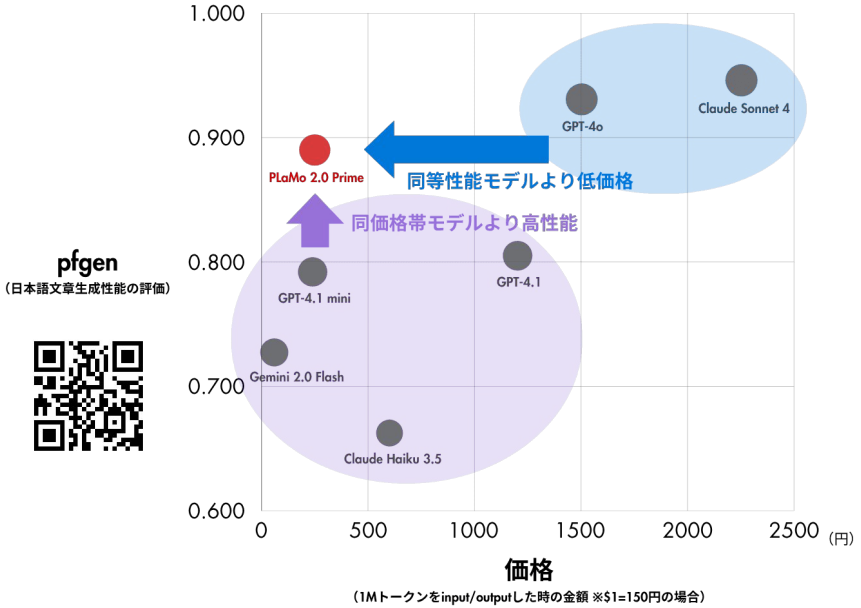
- RAG やプロンプト最適化、ファインチューニングなど、専門家が用途に応じて手厚くサポート

https://plamo.preferredai.jp/

Preferred Networks

PLaMo 2.0 Prime, PLaMo 翻訳

- **性能密度を 10 倍以上改善** [1] 、軽量でフロンティアモデルに匹敵する性能を実現
- PLaMo 翻訳はオープン 8B モデルを公開、**商用・非商用問わず無償利用可** [2]



[1] Densing Law of LLMs, C. Xiao, et al., https://doi.org/10.48550/arXiv.2412.04315

[2] 個人または年商10 億円未満の企業に限る

https://translate-demo.plamo.preferredai.jp/

PLaMo 事前学習モデル開発

2023.8 - 9

PLaMo 13B [1]

- LLaMA arch. base
- Apache 2.0 license
- ABCI 2.0, A100

2024.2 - 5

PLaMo 100B [2]

- Llama 2, 3 arch. base
- PLaMo 100B license
- Google Cloud, H100

2024.10 - 2025.4

PLaMo 2 [2]

- SAMBA base
- Apache 2.0 (1B)
- PLaMo community license (8B)
- 自社クラスタ , H100

2025 後半以降

PLaMo 次世代モデル

- ?

[1] ABCI 第 1 回 大規模言語モデル構築支援プログラムの支援を受け実施
[2] METI および NEDO が行う「GENIAC (Generative AI Accelerator Challenge) 」第 1 期、第 2 期の支援を受け実施

02. LLM 事前学習の難しさ

LLM 事前学習の難しさ

- 従来のモデル学習に比べて必要計算量が桁違い
  - どれだけ高速化しても、**30 日が 1 日にはならない**（今の技術では）

- **LLM 開発は故障との戦い** [1]
  - 事前学習は大量の計算リソースを長時間使う
  - MTBF は計算リソース量に反比例

[1] Meta, https://arxiv.org/abs/2410.21680 (HPCA2025)

モデル	事前学習時間
PLaMo 13B	約 1 ヶ月 x 24 時間
PLaMo 100B	約 3 ヶ月 x 24 時間
PLaMo 2 8B	約 1 ヶ月 x 24 時間
PLaMo 2 30B	約 2 ヶ月 x 24 時間

疲弊しない長期事前学習

約 90 日にわたって不眠不休で監視と保守が必要なら、**チームは疲弊する**
事前学習中は**終わった後の準備** もしたい

発生しうる問題に対処するために以下を実装・整備した

クラスタに依存しない
学習ログの監視

学習中のモデル評価

学習データセットの
完全性保証

学習スナップショット
保存コストの削減

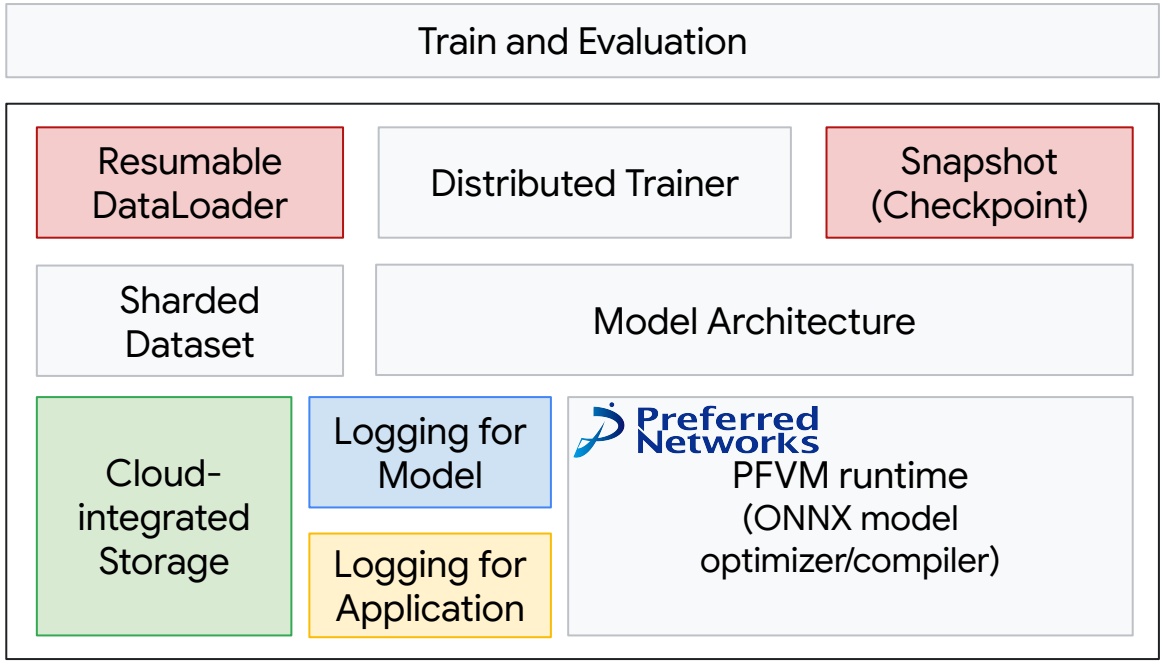
故障時の復旧時間を短縮

注) 計算の高速化には触れません

03. 長期間学習の課題と解決

長期学習の課題と解決

- クラスタに依存しない学習ログの監視
- 学習中のモデル評価
- 学習データセットの完全性保証
- 学習スナップショット保存コストの削減
- 故障時の復旧時間を短縮



クラスタ非依存に学習ログを監視する

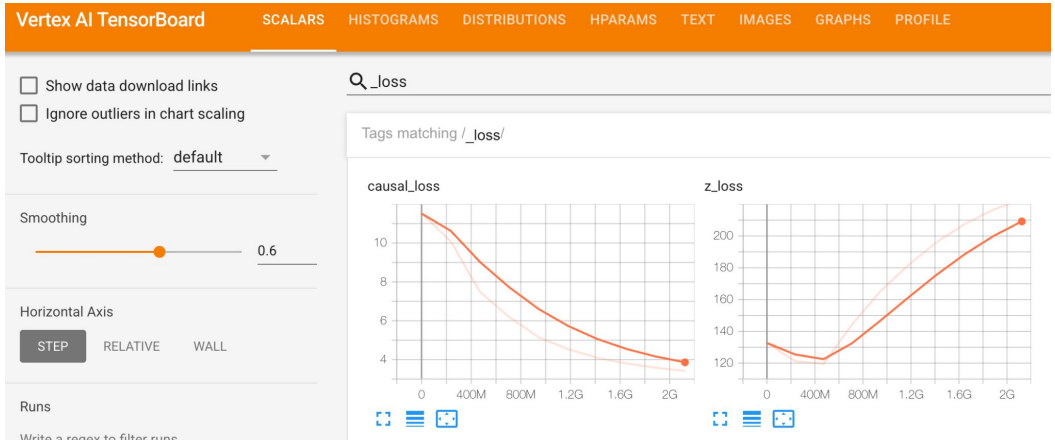
- Python 標準 logging と連携し、Cloud Logging へログを書き込み
- ストリーミングにより**リアルタイムでログを監視**
- ユーザー定義ラベルを使って**ワークロードや MPI ランク単位でログ検索可能**

```python
import google.cloud.logging as glogging
import logging

handler = glogging.handlers.CloudLoggingHandler(
    client=glogging.Client(),
    name="pretrain",
    labels={
        "workflow_name": "yhirokawa-train-exp-20250626-103242",
        "rank": str(torch.distributed.get_rank()),
    },
)

logger = logging.getLogger("pfnet")
logger.addHandler(handler)
logger.info("write to cloud logging")
```

学習中のモデルを評価する

- 学習中 Vertex AI Experiments へメトリック送信、TensorBoard で閲覧
  - Learning curve やハイパーパラメータを**チーム全員で確認できる**

- 学習スナップショットからモデルを構築し評価ワークロードを自動実行
  - 学習済みトークンと評価値をグラフ化し**性能劣化を検出**



学習データセットの完全性を保証する

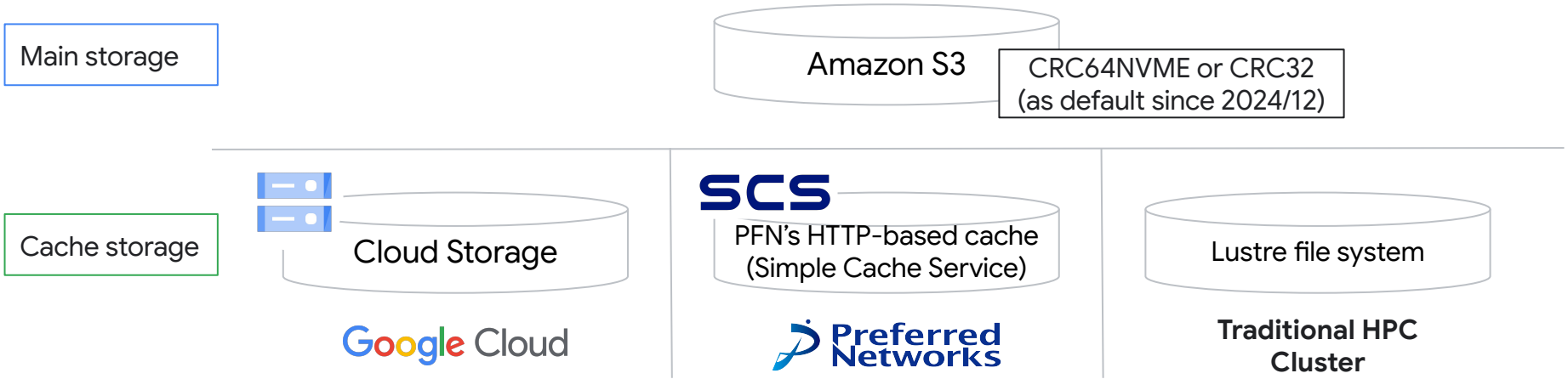
- PFN では LLM の研究開発について自社クラスタで実施
  - 必要な計算リソースと自社クラスタのキャパシティから、プロダクションモデル開発は別のクラスタで実施

- TiB 規模になる巨大な**事前学習データセットの管理が大幅なコスト** に
  - 対象クラスタの共有ストレージへの事前転送
  - 転送したデータセットが
    - 破損していないか
    - 更新前のデータになっていないか
    - **どのシステムでも同じデータが使われているか**

データ検証済みキャッシュ ストレージ

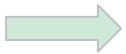
- Cloud Storage や Amazon S3 が持つチェックサムでオブジェクトを検証
  - キャッシュ ストレージ導入により S3 からのダウンロード コストを削減

- インターネットに繋がるシステムなら**データ完全性を保証可能**
  - データの**事前転送不要** (事前学習中の on-the-fly ダウンロードで十分)

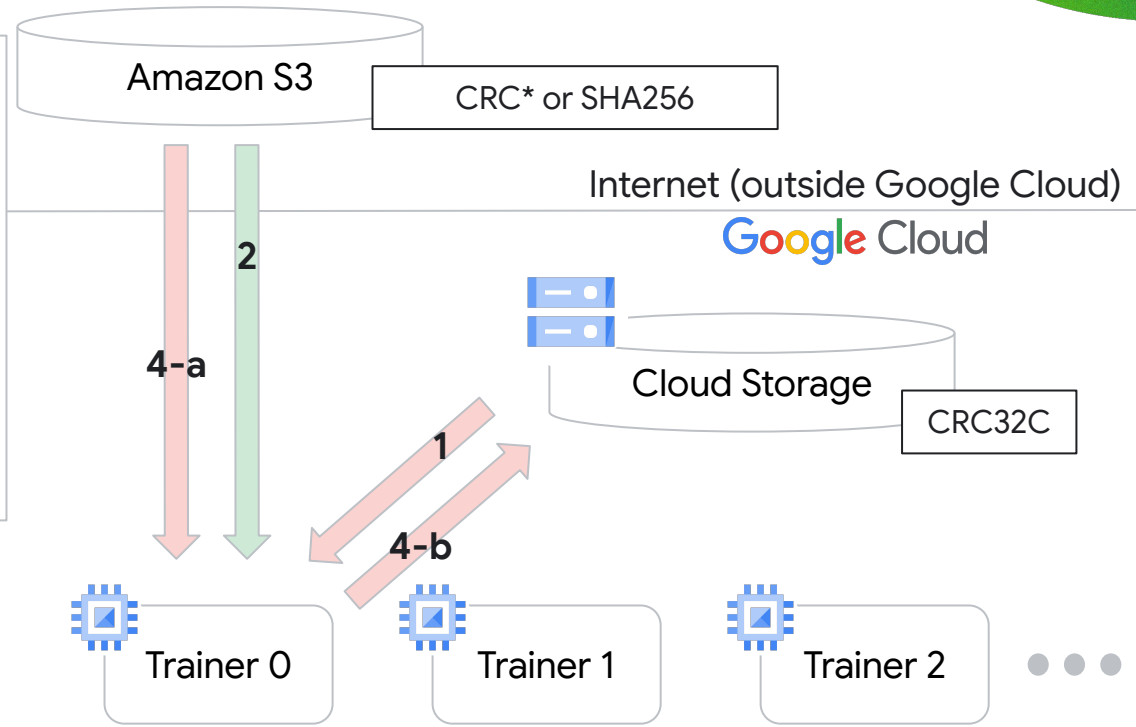


データ取得の流れ

1. キャッシュ ストレージからデータを取得
2. メイン ストレージからチェックサムを取得
3. キャッシュ データのチェックサムを計算
4. チェックサムが**不一致** or キャッシュなし
   a. メイン ストレージからデータを取得
   b. キャッシュ ストレージへ保存
5. チェックサムが**一致**
   a. キャッシュ データを利用

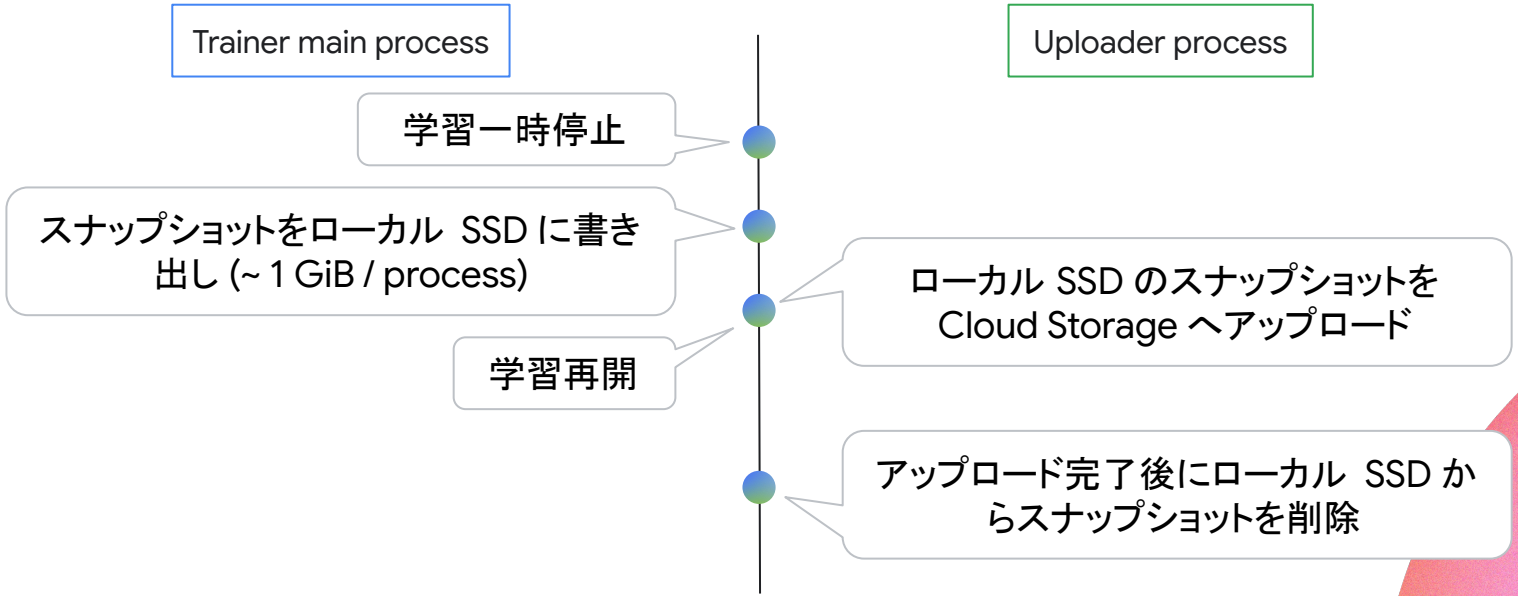

Object metadata (KiB order)


Object data (100 MiB - 1 GiB)



学習スナップショットの保存コストを削減する

- スナップショットの作成は計算を一時停止する必要がある = **学習完了が遅延**
  - ローカル SSD へのステージングと非同期アップロードを実装



停止時間削減の効果

	データ ステージング	Cloud Storage へのアップロード	1 回あたりの計算停止時間
(1)	なし	学習メイン プロセス	5 分以下
(2)	ローカル SSD	専用サブ プロセス	1 分未満 (**約 4 分削減**)

- 2 時間に 1 回の間隔でスナップショット保存をしたい
- (1) のままで学習を進めた場合、+4 分の停止時間が必要
  - 1 日学習すると... 約 50 分、計算停止
  - 90 日学習すると... 約 4 日、計算停止
- **GPU インスタンスの利用コストを 4 日分削減**

故障時の復帰にかかる時間を短縮する

- PyTorch DataLoader の非同期バッチ作成は状態を保存できない
  - 故障前の状態に戻すのは困難、ナイーブな実装では1時間以上必要
  - 状態の保存と復元が可能かつ非同期バッチ作成可能な DataLoader を実装
- **障害発生から学習再開までの時間を約　30 分に削減**

```
loader = DataLoader(ds, batch_size=16, prefetch_size=4, concurrent=True)
it = iter(loader)
for _ in range(5):
    next(it)

state = pickle.dumps(loader)
restarted_it = pickle.loads(state)
assert pickle.dumps(next(it)) == pickle.dumps(next(restarted_it))
```

メンテナンス イベントの捕捉

- Compute Engine インスタンスはメンテナンスによる再起動が不定期に発生
  - 予防的再起動やドライバ更新などと推察

- GPU インスタンスはライブマイグレーション不可
  - イベントを捕捉し、再起動の約 15 分前に学習スナップショットを作成
  - 再起動から復帰まで**約 2.5 時間分**の計算時間を喪失 ⇒ **約 45 分に圧縮**
    - 2 時間に 1 回の頻度、最悪ケースはスナップショット作成直前に再起動

```
gce-gpu-instance$ curl \
  http://metadata.google.internal/computeMetadata/v1/instance/maintenance-event \
  -H "Metadata-Flavor: Google"
# イベントがスケジュールされていない場合
NONE
# イベントが1時間以内にスケジュールされている場合
TERMINATE_ON_HOST_MAINTENANCE
```

04. まとめ

実現したこと

- 長期事前学習に耐えられる仕組み
  - ロギングや実験管理、学習データの完全性保証
  - 学習スナップショット保存や故障復帰コストの削減

- 1 日に発生する計算の停止時間のうち約 2 - 3 時間を取り戻せた
  - 学習スナップショット保存　　約 50 分
  - 故障発生から学習再開まで　　約 30 分 (またはそれ以上)
  - メンテナンス イベントからの復帰　約 2 時間

- なにより**チームが疲弊することなく 3 ヶ月にわたる事前学習を遂行できた**
  - 学習完了後の様々な準備や開発作業も進められた

おわりに

- **LLM 開発は故障との戦い**
- 計算の高速化と**同時に**故障への備えが非常に重要
  - どちらも最後には開発費の削減になる
- PFN は今後も競争力あるフルスクラッチ モデルの開発と提供に努めます

https://plamo.preferredai.jp/