CNCF AI ワーキング グループ

クラウドネイティブ人工知能

著者

Adel Zaalouk
Alex Jones
Andrey Velichkevich
Boris Kurktchiev
Cassandra Chin
Cathy Zhang
Claudia Misale
Huamin Chen
Joel Roberts
Kai-Hsun Chen
Malini Bhandaru
Michael Yao
Nikhita Raghunath
Peter Pan
Rajas Kakodkar
Rasik Pandey
Ricardo Aravena
Ronald Petty
Ryan Taylor
Saad Sheikh
Shawn Wilson
Tom Thorley
Victor Lu

CLOUD NATIVE COMPUTING FOUNDATION

出版

2024 年 3 月 20 日（第 1 版）

要旨

クラウド ネイティブ（CN）と人工知能（AI）は、今日最も重要な技術トレンドです。クラウド ネイティブ[1]技術は、アプリケーションを実行するためのスケーラブルで信頼性の高いプラットフォームを提供します。AI と機械学習（ML）の最近の進歩を考えると、AI/ML は着実にクラウドのワークロードの主流になりつつあります。CN 技術はある範囲においては AI/ML ワークロードをサポートしている一方で、課題やギャップが残っており、イノベーションやより良い対応をする余地があります。

本稿では、最先端の AI/ML 技術の概要を紹介し、次に CN 技術が提供しているものを紹介します。本稿は、変化するクラウド ネイティブ人工知能（CNAI）のエコシステムと今後の発展の機会（チャンス）を理解するための知識をエンジニアとビジネス担当者に提供します。

読者の背景や関心に応じた項目を読み進めてください。マイクロサービス[2]や Kubernetes(K8s) などの CN 技術[3]に触れていることを前提としています。AI システムのエンジニアリング経験がない方には、最初から最後まで読むことをお勧めします。また、AI/ML の導入や提供が更に進んでいる方には、ユーザー ペルソナ[4]に従って、現在取り組んでいる、あるいは解決に関心のある課題に関連するセクションに飛ぶことをお勧めします。また、このコンテキストでは社会がどこに投資すべきかも共有しています。

目次

クラウド ネイティブ人工知能（CNAI）入門

クラウド ネイティブと AI 技術の融合である CNAI に入る前に、それぞれの進化を簡単に検証しておきましょう。

クラウド ネイティブの出現

2013 年以来広く知られ[5]、使用されているクラウド ネイティブ（CN）という用語は、LXC[6]から Docker[7]、Kubernetes(K8s)[8]にいたるコンテナ技術の台頭とともに人気が高まりました。今日、クラウド ネイティブは、より広義には、再利用性の高いモジュール設計と開発を促進するマイクロサービス設計パターンを使用して構築されたバランスの取れたシステムを目指すものであり、デプロイアビリティ、スケーラビリティ、耐障害性にも適しています。

Cloud Native Computing Foundation は、クラウド ネイティブを次のように定義しています[9]

クラウド ネイティブ技術は、パブリック、プライベート、ハイブリッド クラウドなど、最新の動的環境でスケーラブルなアプリケーションを構築し、実行できるようにするものです。コンテナ、サービス メッシュ、マイクロサービス、不変のインフラ、宣言型 API が、この技術の一例です。

これらの技術は、柔軟性があり、管理可能で、観測可能な疎結合システムを実現します。ロバストな自動化と組み合わせることで、エンジニアは最小限の労力で、インパクトの大きな変更を迅速かつ予測通りに行うことができます。

Cloud Native Computing Foundation は、オープンソースでベンダーニュートラルなプロジェクトのエコシステムを育成・維持することで、このパラダイムの採用を推進しています。私たちは最先端のパターンを民主化し、これらのイノベーションを誰もが利用できるようにしています。

クラウド ネイティブ人工知能は、クラウド ネイティブの進化した延長線上にある

クラウド ネイティブ人工知能（CNAI）とは、クラウド ネイティブの原則を利用して AI アプリケーションとワークロードを構築・展開するためのアプローチとパターンを指します。AI に特化した反復可能でスケーラブルなワークフローを実現することで、AI 実務者は自分の専門分野に集中することができます。

Kubernetes は、プライベート クラウド、パブリック クラウド、ハイブリッド クラウドを提供する、事実上のクラウド オペレーティング システムへと進化しました。Kubernetes は、複数のタイプのネットワーク、ストレージ、および計算資源を処理する分散オーケストレーターを実装しています。さらに K8s は、GitOps のような DevOps[10]のベストプラクティスを利用できるインターフェイスを提供しています[11]。すべてのクラウド サービス プロバイダー（CSP）は、サービスとして Kubernetes のいくつかのフレーバーを持っており、AI/ML を含むさまざまなワークロードを実行するためのインフラや多数のサポート サービスへのアクセスを容易にしています。

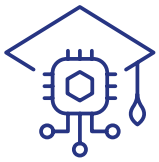
人工知能の進化

1956 年に始めて用語として導入された人工知能[12]とは、人間の知能をシミュレートする機械の能力のことです。数十年にわたり、音声認識、機械翻訳、画像処理、ゲームプレイなどのアプリケーションで使用されています。さらにはジョパディ プレイヤーとしても活躍しています[13]。しかし、人工ニューラル ネットワークとディープ ラーニングの技術革新のおかげで、AI はより最近、主に自然言語理解に応用され、爆発的な知名度を獲得しました。AI には主に、識別型（discriminative）と生成型（generative）の 2 つに分類されます。

識別 AI は、判断の境界や分類を学習しようとするもので、その知識は「モデル」として取り込まれ、新しいデータを予測するために使われます。例えば、電子メールをスパムかそうでないかに分類したり、猫と犬の画像を区別したり、その他いろいろなことができます。識別 AI は通常、（機械学習の一種である教師あり学習によって）望ましい出力がわかっているタスクに使用されます。AI はシーケンス予測に優れており、例えば、私たちの個人的な書き方の情報を含む既存の大量のテキストを分析することで、私たちが次に何を入力するかを高い確率で推測することができます。

生成 AI は、データ内に潜在する構造や表現を学習します。例えば、言葉のプロンプトから物語や音楽、ビジュアル アートを創作するように、これらの構造や表現を使って新しいデータを合成することができます。生成 AI は、望ましい出力が未知であったり、「正しい」出力が定義されていないようなタスクに使用されます。生成 AI によって、AI は人間が創造的、独創的、崇高と考えるものへと超越しました。AI の目を見張るようなブレークスルーのいくつかを詳しく見てみましょう。

畳み込みニューラル ネットワーク（CNN）[14] は 1980 年代に初めて開発されましたが、広く使われるようになったのは 2000 年代初頭のことです。近年、CNN は大規模な画像データセットから学習する能力を持ち、物体検出、画像分類、セグメンテーションなど、さまざまな画像処理タスクで優れた性能を発揮することから、ますます人気が高まっています。



トランスフォーマー は 2017 年にトロント大学とグーグルの研究者によって開発されました。トランスフォーマーは、スケールド ドットプロダクト アテンション（scaled dot-product attention）と呼ばれる特殊なメカニズムを使用しており、記憶のような構造を持っています[15]。トランスフォーマーに基づくモデルは、質問への回答、テキストの要約、翻訳などの自然言語処理タスクに非常に効果的です。そのため、ほとんどの大規模言語モデル（Large Language Models：LLM）には欠かせない存在です。最も有名な LLM は GPT であり、人気のある ChatGPT サービスを支えているモデルです[16]。

LLM は膨大なデータセットで訓練されます。LLM は、長いプロンプトのシーケンスを受け取り、文脈に応じた応答を生成します。さらに、時事問題、医学、法律など、追加データを用いて専門的な領域向けに微調整することも可能です。人間のフィードバックからの強化学習（RLHF）や直接優先最適化（DPO）のような微調整のための新しい技術は、LLM をさらに魅力的なものにするために開発されました。

研究とイノベーションにより、エンドユーザーとのインタラクションはかつてないほど速く、より創造的で、より正確なものとなりました。データ サイエンスとソフトウェアのイノベーションと同様に重要なのは、モデル推論（AI モデルから結果を計算するプロセス）とモデル トレーニング（データから AI モデルを構築するプロセス）を強化するインフラの進化です。AI アクセラレーター技術を使えば、AI 実務者はより高速に反復作業を行い、数か月だった作業が数日や数週間で、より高品質なモデルを提供することができます。さらに、データ サイエンティストや統計学者が採用してきたいくつかの伝統的な技術は、CN システムの能力を活用するために再評価されつつあります。

クラウド ネイティブと人工知能の融合

前のセクションで述べたように、AI とは、人間と同様のタスクを実行できるシステムを作り出すことを目的とした、より広い概念です。機械学習は、データから学習し、情報に基づいた予測や意思決定を行う方法です。機械学習とは、データから学習し、データに基づいた予測や意思決定を行う方法であり、明示的なプログラミングを行うことなく、アルゴリズムを使用して学習し、時間をかけて改善します。これは自動化の一形態と考えることができます。最後に、データサイエンスは学際的な分野として、統計学、数学、コンピュータサイエンスの技術を融合し、データの分析や解釈から機械学習アルゴリズムの適用まで、幅広い活動を実施します。

大まかに考えると、AI、ML、データサイエンスのアプリケーションは、**予測 AI** と **生成 AI** の 2 つに大別できます。予測 AI は、既存のパターンや結果を予測や分析することを目的としています（例えば、分類、クラスタリング、回帰、物体検出など）。対照的に、生成 AI は、新しく独創的なコンテンツを生成することを目的としています（LLM、RAG[17] など）。このように、予測 AI と生成 AI を支えるアルゴリズムと技術には大きな違いがあります。

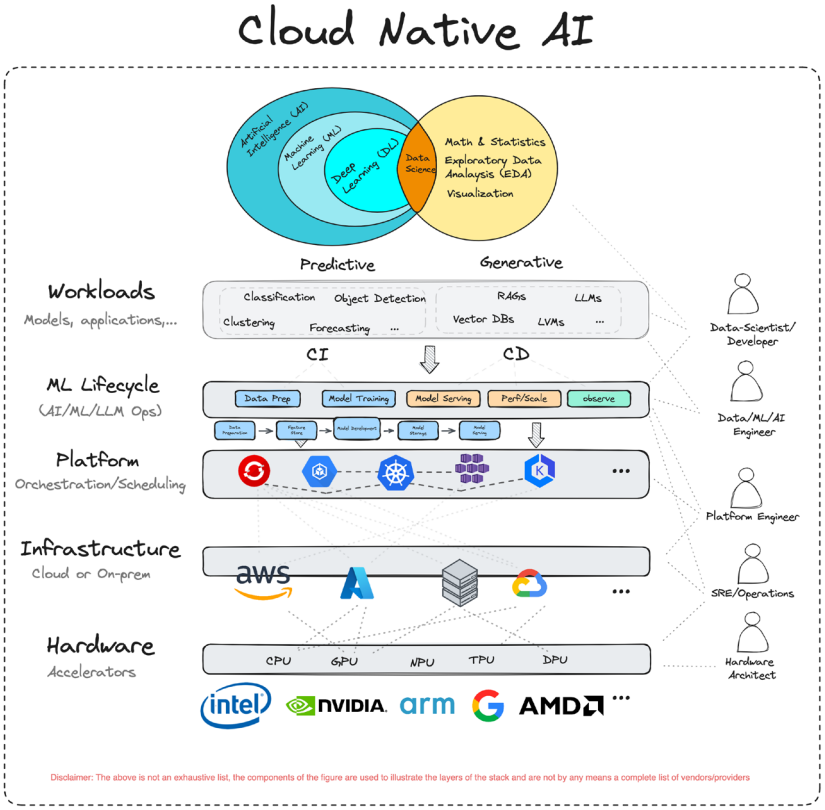


図 1
クラウド ネイティブ AI

以下は、予測 AI と生成 AI が、コンピューティング、ネットワーキング、ストレージの各分野で異なるニーズを検出した分野の一例です。

課題 / ニーズ	生成 AI	予測 AI
計算力	非常に高い 専用のハードウェアが必要	中程度から高い 汎用のハードウェアで十分
データ量と多様性	トレーニング用の膨大で多様なデータセット	予測のための具体的な過去のデータ
モデルのトレーニングと微調整	専門的な計算による複雑な反復トレーニング	適度なトレーニング
拡張性と柔軟性	拡張性と柔軟性の高いインフラ（可変性があり集中的な計算が要求される）	拡張性は必要だが、柔軟性の要求は低い バッチ処理またはイベント駆動タスク
ストレージとスループット	優れたスループットの高性能ストレージ 多様なデータタイプ データへのアクセスには高いスループットと低遅延が要求される	中程度のスループットで効率的なストレージ データ分析に重点を置き、データ生成はあまり行わない
ネットワーキング	データ転送とモデル同期（分散型トレーニング中など）のための高帯域幅と低遅延	データアクセスのための一貫した信頼性の高い接続性

これからのセクションでは、どちらの形でも生じるニーズに応える方法と、それに伴う課題、そしてそうした課題に直面した際に採用できる方法について探っていきます。

クラウド ネイティブ人工知能とは何か？

クラウド ネイティブ人工知能は、AI ワークロードを展開、実行、拡張するための実用的なシステムの構築を可能にします。CNAI ソリューションは、AI アプリケーション科学者、開発者、デプロイヤーが、クラウド インフラ上で AI ワークロードを開発、デプロイ、実行、スケーリング、モニタリングする際に直面する課題に対処します。基盤となるクラウド インフラのコンピューティング（CPU や GPU など）、ネットワーク、ストレージ機能を活用し、分離しつつ制御できる共有メカニズムを提供することで、AI アプリケーションのパフォーマンスを加速し、コストを削減します。

図 2（下図）は、これらの実現メカニズムをツールと技術の観点でマッピングしたものです。

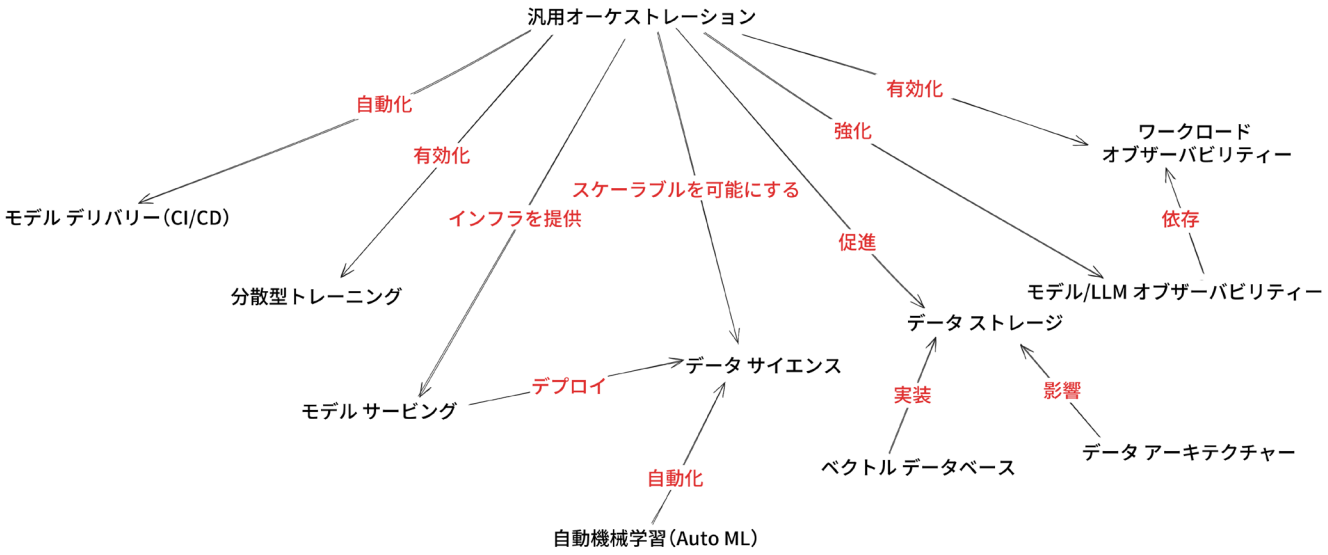


図 2
利用可能なツールや技術 [18]

クラウド ネイティブ インフラで AI を動かす

クラウド サービス プロバイダーや AI 企業が発表したメディア記事では、AI にとってのクラウド ネイティブの価値が強調されています [19] [20]。この分野においてクラウド プロバイダーやスタートアップによる AI 関連サービスが登場したことは、クラウド ネイティブの原則が AI の進化に必要なシステムを形作ることができることを強調しています。

OPENAI

Kubernetes を 7,500 ノードに拡張

HUGGING FACE

Hugging Face がマイクロソフトと協力し、Azure 上で Hugging Face モデル カタログ サービスを開始

クラウド ネイティブ人工知能は、クラウド ネイティブの進化した延長線上にある。

Kubernetes は、コンテナのデプロイと管理に使用できるオーケストレーション プラットフォームであり、軽量でポータブルな自己完結型のソフトウェア ユニットです。AI モデルは**コンテナ**にパッケージ化され、K8s クラスタにデプロイされます。コンテナ化は AI モデルにとって特に重要です。異なるモデルは通常、異なる、そしてしばしば矛盾する依存関係を必要とするからです。これらの依存関係をコンテナにより分離することで、モデルのデプロイにおいてはるかに大きな柔軟性が生まれます。CN ツールは、AI モデルの効率的でスケーラブルなデプロイを可能にしました。そして、AI ワークロード専用にこれらを調整する取り組みが続けられています。

Kubernetes スケジューラー [21] は進化を続けており [22][23]、特に AI ワークロードにおいて高速化の分野で高い人気を集めている GPU(Graphics Processing Unit) をより良く統合し、共有することをサポートしています。GPU を共有するアプリケーションをサポートしてマルチテナンシーを処理するだけでなく、Kubernetes 外のリソースのリモート プールを活用するための取り組みも進行中です。

優れた推論を得るための AI モデルの訓練とテストには、高品質のデータが必要です。クラウド ネイティブ インフラは、データ レイクやウェアハウスなど様々な方法でデータにアクセスできます。多くのクラウド プロバイダーは、ブロック、オブジェクト、ファイル ストレージ システムを提供しており、低コストでスケーラブルなストレージを提供するのに最適です。例えば、モデルのサイズは数ギガバイトに達することもあります。トレーニング段階では、モデルのチェックポイントを毎回引き出すことにより、ネットワーキングとストレージの帯域幅に大きな負荷をかける可能性があります。コンテナ化されたアーティファクトとしてモデルを扱うことで、OCI[24] レジストリにホスティングでき、キャッシュが可能になります。さらに、アーティファクトへの署名、検証、証明、データ実証など、**ソフトウェア サプライチェーン**のベストプラクティスをモデルに適用することも可能になります。さらに、モデルやアーティファクトをコンテナ化することで、WebAssembly(WASM) バイナリへのバンドルが容易になります。WASM は、プラットフォームに依存しない、効率的な推論への CN アプローチです。

なぜクラウド ネイティブ人工知能なのか?

クラウドは弾力的で常時接続可能なインフラであるため、企業、スタートアップ、開発者は、迅速なプロトタイプの作成、新サービスの提供、ソリューションの拡張などを行うことができようになりました。また、リソースの共有により、コスト効率も向上しています。一般的なユーザーは、ハードウェアの注文や、スペース、電源、ネットワーク接続、冷却、ソフトウェア ライセンス、インストールなどのロジスティクスを心配する必要がなくなりました。AI にも同様の懸念を抱いています。迅速なプロトタイピング、小規模・大規模のトレーニングや推論タスクに取り組むためのストレージ、ネットワーク、コンピューティング リソースへのアクセスについてです。

クラウド ネイティブ システムの改善に AI を活用

ログの自然言語処理（NLP）のために LLM の機能を活用するにせよ、オブザーバビリティー ツールとしてパッケージ化されるにせよ、AI を活用したソリューションやプロジェクトがオペレーターやエンドユーザーの手に渡り、生産性を向上させたり、生活をしやすくしたりしています。そのようなオープンソースの Cloud Native Computing Foundation (CNCF) のプロジェクトの 1 つが K8sGPT です [25]。K8sGPT は、Bedrock、Cohere などの LLM のパターン認識と言語機能を活用し、K8s オペレーターの日常業務を支援しています。しかし、より重要なのは CN と AI の共生が、エコシステムに新たな予期せぬ発展のチャンスをもたらすことです。例えば、複雑なシステムを操作・管理できる技術力の低いユーザーの増加が予想されています。

クラウド ネイティブ人工知能の課題

CNAI の課題はペルソナによって異なることに注意が必要です [26]。また、クラウド ネイティブの柔軟でスケーラブルなプラットフォームは AI ワークロードに有望である一方、AI のスケールとレイテンシーのニーズには課題があり、CN 技術のギャップを露呈すると同時に、新たな発展のチャンスをもたらしています。文献では MLOps[27] とも呼ばれているエンドツーエンドの ML パイプライン [28] に関する文脈でこれらを説明します。時間と空間、並列性、および同期といった伝統的なトレードオフの問題がすべて表面化し、使いやすさが犠牲にさらされます。要約すると、ML ライフサイクルは以下のようになります。

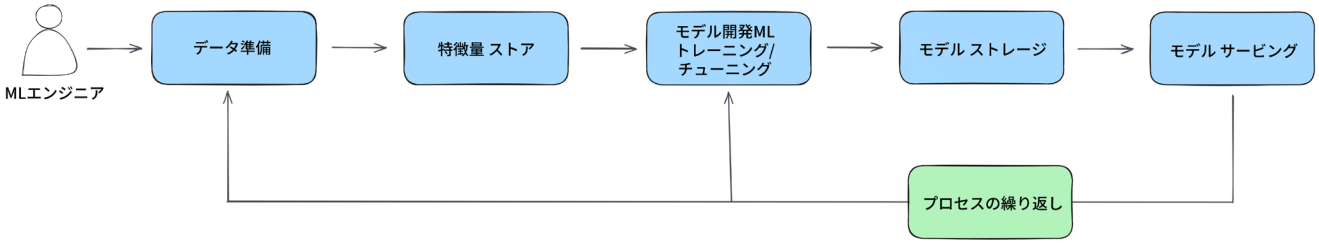


図 3
ML ライフサイクル

典型的な ML パイプラインは以下のような構成になっています。

- データ準備（収集、クリーニング / 前処理、特徴量エンジニアリング）
- モデル トレーニング（モデル選択、アーキテクチャ、ハイパーパラメーター チューニング）
- CI/CD、モデル レジストリ（ストレージ）
- モデル サービング
- オブザーバビリティー（使用負荷、モデル ドリフト、セキュリティ）

特に LLM の場合、学習、類似性検索、モデルサイズに関わるデータ量が多く、それぞれメモリとパフォーマンスを考慮する必要があります。CPU のアクセス制御とスケジューリングは CN が行うものの、GPU の共有時の適切な割り当てはまだ発展途上です。ML のトレーニング段階は探索がすべてであり、中間モデルのパフォーマンスを追跡して、どれを維持し、さらに高い精度を得るためモデルのパラメーターをどのように調整するかを決定する必要があります。取り扱われるデータの機密性とモデルの本質的価値を考えると、セキュリティはより重要です。オブザーバビリティーは、モデルのドリフトや使用負荷などを検出するために不可欠です。各パイプライン段階における課題をもう少し掘り下げてみましょう。読者は、自分のドメインに関連する更なる課題を検討し、会話に加えてみてください [29]。

データ準備

AI/ML パイプラインの最初のフェーズであるデータ準備には様々な課題があります。これらは大まかに 3 つの主要なカテゴリーに分類することができます。大きなデータ サイズの管理、開発時とデプロイ時の確実なデータ同期、データ ガバナンス ポリシーの遵守です。

データサイズ

より優れた AI/ML モデルを構築するためのデータ需要は、ムーアの法則を上回るスピードで増加しており、18 カ月ごとに倍増しています[30]。データ管理 / 取扱い、データ処理、データ分析のいずれにおいても、AI/ML モデル構築のためのデータ需要は急速にエスカレートしています。したがって、分散型クラウド ネイティブ コンピューティングと効率的なデータ移動や保存は、こうした計算需要とハードウェア能力のギャップを埋めるために不可欠となります。

データ同期

データは複数の異なる場所から、異なるフォーマットで入手する必要があるかもしれません。開発環境と本番環境は、多くの場合異なるものであり、これらに加えて、パーティショニングや同期といった分散コンピューティングから生じる複雑性の増大にも対処する必要があります。後者について詳しく見てみましょう。

Spark のようなデータ処理システムでは、業界標準のインターフェースである SQL は、ローカルでプロトタイプを作成する場合でも、大規模なワークロードを分散して実行する場合でも、使い慣れた統一されたエクスペリエンスをユーザーに提供する上で重要な役割を果たしています。しかし、ML ワークロードには業界標準のインターフェースがありません。その結果、データ サイエンティストは、ローカルで小規模なデータセットを使って ML の Python スクリプトを開発し、分散システム エンジニアがこれらのスクリプトを分散実行用に書き換えます。分散された ML ワークロードが期待通りに機能しない場合、データ サイエンティストはローカルの Python スクリプトを使って問題をデバッグする必要があるかもしれません。このプロセスは非効率で、しばしば効果的ではありません。より優れたオブザーバビリティー ツールや、コンテナ技術による再現性があるにもかかわらず、これが事実です。

ローカルの開発環境と本番環境の間のこの矛盾を解決するための、実行可能な解決策は存在します。1 つ目は、エンドツーエンドの ML ライフサイクルをサポートする業界標準のインターフェースを使用することです。例えば、ユーザーは PyTorch や TensorFlow のようなネイティブの ML フレームワークの API を活用して学習コードを作成し、Python ランタイムでローカルに実行して検証することができます。その後、ユーザーは簡単に同じコードを再利用し、Kubeflow の Python SDK を活用して、Kind/Minikube を介して分散方式を使ってローカルでこのコードを実行したり、同じ Python SDK を使用してリモートの大規模 Kubernetes クラスタにデプロイすることで、同様に簡単にトレーニング コードを拡張させることができます。もう一つの選択肢は、Ray のような汎用分散コンピューティング エンジンを使用することです。抽象化により、ユーザーは同じ Ray スクリプトをローカル環境と本番環境でシームレスに実行することもできます。

データ量は広範囲にまたがる問題です。それはトレーニング段階でも顕在化します。

データ ガバナンス

データ ガバナンスは、信頼を築き、責任ある AI 開発を保証するために極めて重要です。データ ガバナンスに関しては、3 つの重要な柱を考慮する必要があります。

1. **プライバシーとセキュリティ：**GDPR[31] や CCPA[32] のような複雑なデータ プライバシー規制を乗り切ることが不可欠です。AI モデルで使用される機密データを保護するために、強固なセキュリティ対策を実施する必要があります。暗号化、アクセス制御、定期的な脆弱性評価により貴重な情報を保護する必要があります。
2. **所有権とリネージ：**AI のライフサイクルを通じて、データの収集から使用まで、データの所有者とアクセス権を明確に定義することが不可欠です。データの系統追跡（Data lineage tracking）ツールを活用して、データがシステム内をどのように流れているかを把握し、透明性と説明性を確保する必要があります。そうすることで、機密情報への不正アクセスや悪用を防ぐことができます。

3. **バイアスの軽減：**AI モデルは、学習させたデータと同程度の性能しか持ちません。したがって、データとアルゴリズムに潜在するバイアスを積極的に監視し、対処することが不可欠です。例えば、多様なデータセットを使用すること、公平性の評価基準を採用すること、モデルの限界を把握することなど、公正で倫理的な結果を確実に出すためにモデルを継続的に評価することなどが含まれます。モデル カード [33] は、これらを把握するために進化しています。

データのプライバシーとセキュリティは、あらゆる段階での配慮を必要とする横断的な問題です。

モデル トレーニング

モデル トレーニングのデータ量は指数関数的に増加しており、その結果、さらなる並列性を達成するための分散処理とアクセラレーターの必要性が生じています。さらにトレーニングは反復的な複数ステップのプロセスであるため、スケーリングは複雑な複数コンポーネントの調整タスクとなります。このセクションでは、これらの側面について詳しく説明します。

高まる処理要求

LLM は、AI/ML のトレーニングや推論コンピューティングの需要の高まりに対応するため、急速に限界に挑戦しており、アクセラレーターが普及しています。これらのアクセラレーターは、さまざまな機能を持つ複数のベンダーの GPU から、Google のテンソル処理ユニット（TPU）、Intel の Gaudi、さらにはフィールド プログラマブル ゲート アレイ（FPGA）まで多岐にわたります。これらの多様な計算リソースには、仮想化サポート、ドライバ、コンフィギュレーションと共有機能、CN スケジューラーの機能強化が必要です。さらに、これらのアクセラレーターの可用性とコストは限られているため、マルチクラウド リソース バンディングや、sky[34] コンピューティングの探求も促されています。

CN 技術を AI に使用する場合、GPU の仮想化と動的割り当てが複雑になる可能性があります。vGPU、MIG、MPS（用語集を参照）、Dynamic Resource Allocation(DRA)[35] などの技術は、ポッド内のコンテナ間の分離と共有を提供しながら、複数のユーザーが単一の GPU を共有することを可能にします。これらは、複数のワークロードが同時に GPU の恩恵を受けられるようにするだけでなく、GPU の利用率を高めてコストを削減することができます。しかし、実装には慎重なオーケストレーションと管理が必要であり、特にリソースを動的に割り当てたり、割り当てなかったりする場合には注意が必要です。スムーズで効率的な統合を実現するには、AI チームと CN エンジニアリング チームの緊密な連携が必要です。

コスト効率

クラウド ネイティブ環境特有の柔軟性と拡張性により、組織は変動する需要に基づいて動的にリソースを供給し、拡張することができます。この側面は AI タスクにも当てはまります。しかし、変化するワークロードの需要に対応するためのリソースの適切なサイジングとリアクティブなスケジューリングは、高価で供給量が限られている GPU などのアクセラレーターにおいては、さらに切実です。GPU をより有効に活用するために、**GPU を分割できるようにする**必要があるのです。

モデル配信中のカーボン フットプリントの削減は、需要に基づいて動的にリソースを調整する自動スケーリング配信フレームワークを使用して達成することができます [36]。LF AI&Data Foundation のプロジェクトである KServe[37] は、そのような機能を提供しています。持続可能性 [38] は、より小さく、より専門化された混合専門家モデル（mixture of experts）の使用、圧縮や蒸留のような技術など、様々な手段によって大幅に改善することができます。地理的に、ML サービングを再生可能なエネルギー源やよりクリーンなエネルギー源から電力を供給される地域に分散させることで、二酸化炭素排出量を大幅に削減することもできます [39]。

ML モデルの責任ある開発には、カーボン フットプリントに関するメタデータを含めることで、モデルの排出が環境に与える影響の追跡と報告を支援することができます。 *mlco2*[40] や *codecarbon*[41] のような追加ツールは、制限はあるものの、物理的なトレーニングの前に新しいニューラル ネットワークのカーボン フットプリントを予測するのに役立ちます。

拡張性

AI/ML のワークフローは複雑で、分散環境で実行される多様なコンポーネントによって特徴付けられます。トレーニングにおいては、扱われるデータ量の多さと、モデルが収束まで複数回のトレーニングをサポートする必要があることによって、この複雑さが特に悪化します。それぞれが特定の AI 機能をカプセル化したさまざまなマイクロサービスの規模を調整するには、シームレスな通信と同期を確保するための複雑なオーケストレーションが必要です。さらに、AI モデルとフレームワークの異種性が標準化を複雑にし、さまざまなアプリケーションに適用できる汎用的なスケーリング ソリューションの作成を困難にしています。

オーケストレーション / スケジューリング

前述したように、クラウド ネイティブ ツールやプロジェクトは、コンテナ化、マイクロサービス、スケーラブルなクラウド インフラの固有の機能を活用することで、AI ワークロードのオーケストレーションとスケジューリングを簡素化しています。複雑な AI ワークフローをモジュール化されたコンポーネントに分解できるため、特定の機能を独立して管理・拡張することが容易になります。

しかし、前述したように、GPU は貴重で需要の高いリソースであり、GPU ベースの AI ワークロードの共有とスケジューリングをより効率的に管理する能力は、AI 開発チームの成功に不可欠です。ビン パッキング、配置、リソース競合、先取りなどの高度なスケジューリング ニーズに対処するための十分にテストされたツールは、クラウド ネイティブ AI が成功するために不可欠であり、基礎となります。Kubernetes では、Yunikorn[42]、Volcano[43]、Kueue[44] といった取り組みを通じて、より優れたスケジューリング サポートが進化しており、後者 2 つはバッチ スケジューリングに対応しています。トレーニング ジョブは、ギャング（またはグループ）スケジューリング[45] の恩恵を受けます。ジョブに属するコンテナ レプリカが正しく機能するためには、オールオアナッシングの配置ポリシーが必要であり、これらのジョブは簡単にスケール アップまたはスケール ダウンできないからです。ギャング スケジューリングのサポートには、新たな発展のチャンスが広がっています。

依存関係の管理

AI アプリケーションは多くの場合、特定のフレームワークやライブラリのバージョンに依存しており、これらの依存関係は標準的なコンテナ イメージではすぐに利用できなかったり、互換性がなかったりします。

多くの AI ワークロードは GPU アクセラレーションの恩恵を受けるため、GPU 上でのワークロード実行をサポートするために必要な GPU ドライバやライブラリを用意することは、特に異なるベンダーや GPU アーキテクチャを扱う場合、困難な場合があります。例えば、NVIDIA デバイス上で分散型トレーニングを実行する場合、最適化されたマルチ GPU およびマルチノード通信プリミティブを利用するために、NVIDIA Collective Communications Library(NCCL) を使用することができます。ライブラリのバージョンが異なると、パフォーマンスが異なる可能性があります。ビルドが再現可能であることは、すべてのソフトウェアにとってのお手本であり、ランタイムの互換性がなかったりパフォーマンスが悪化するのを避けるために、バージョン依存関係を使用する必要があります。

モデル サービング

モデル サービングは、負荷の変動や多くの場合レイテンシーが要求されるため、主にデータ処理やトレーニングとは異なります。さらに、コスト削減のためにインフラを共有することに加えて、サービスの回復力も考慮しなければなりません。また、AI モデルの特性はそれぞれ異なり、古典的な ML、ディープ ラーニング（DL）、生成 AI（GAI）LLM、さらに最近ではマルチモーダル アプローチ（テキストからビデオなど）によっても大きく異なります。作業負荷が異なれば、ML インフラから様々なサポートが必要となります。例えば、LLM が登場する以前は、モデルの処理に必要な GPU は 1 つだけでした。ワークロードがレイテンシーに敏感でなければ、CPU ベースの推論を選択するユーザーもいました。しかし、LLM を提供する場合、トランスフォーマー デコーダーの自己回帰的な性質により、性能のボトルネックが計算依存からメモリ依存にシフトしています[46]。

このセクションでは、CN がこれらの側面をどのように支えているのか、またどのような課題が残っているのかを探ります。

マイクロサービス アーキテクチャと開発者の体験

CN はマイクロサービス アーキテクチャに基づいています。しかし、これは AI にとっては、ML パイプラインの各ステージを個別のマイクロサービスとして扱うことが課題となるかもしれません。多くのコンポーネントがあることで、その出力とハンドオフの維持と同期が困難になる可能性があります。ユーザーがラップトップ上でこれらのソリューションで遊びたいだけだとしても、何十ものポッドを作成する必要があるかもしれません。この複雑さにより、インフラは多様な ML ワークロードに適応する柔軟性を欠くことになります。

第二に、マイクロサービスベースの ML インフラは、断片的なユーザー エクスペリエンスをもたらします。例えば、AI 実務者は日々のワークフローの中で、ML の Python スクリプトだけに集中するのではなく、コンテナ イメージを構築したり、カスタム リソースの YAML ファイルを書いたり、ワークフロー オーケストレーターを使ったりする必要があるかもしれません。この複雑さは学習の難しさとしても現れ、ユーザーは自分の専門外または興味外の多くのシステムを学ぶ必要があります。

第三に、ML モデルのライフサイクルで異なるシステムから各ステージを統合する場合、コストは大幅に増加します。Samsara のエンジニアリング ブログ [47] によると、同社の ML プロダクション パイプラインは、データ処理、モデル推論、ビジネス ロジックの各ステップが別々の複数のマイクロサービスにまたがってホストされていました。インフラが分割されているため、リソースを同期させるための複雑な管理が必要となり、開発とモデル リリースのスピードが低下していました。その後、Samsara は Ray を使用して、本番 ML パイプラインのパフォーマンスを向上させる統合 ML プラットフォームを構築し、主にリソースの共有とステージ間のシリアライズとデシリアライズの排除によって、同社の年間 ML 推論コストの合計を 50%近く削減しました。

これらの問題は、Ray のような汎用分散計算エンジンをベースとした統合 ML インフラの必要性を浮き彫りにしています。Ray は既存のクラウド ネイティブ エコシステムを補完し、計算に集中することで、クラウド ネイティブ エコシステムはデプロイとデリバリーに集中することができます。Ray/KubeRay コミュニティは、Kubeflow[48]、Kueue[49]、Google GKE[50]、OpenShift[51] など、複数のクラウド ネイティブ コミュニティと広範囲に協力しています。

モデル配置

ユーザーが理想とするのは、単一のクラスタに推論用の複数のモデル（場合によっては関連性のないモデル）を配置することであり、同時に推論フレームワークを共有することでコストを削減し、モデルの分離を図ることです。さらに、耐障害性を確保するために、異なる障害ゾーンにレプリカを配置したいとも考えています。Kubernetes は、異なるトポロジー ドメイン（ゾーン、ノードなど）でワークロードをスケジューリングするためのアフィニティ メカニズムとアンチ アフィニティ メカニズムを提供しています [52]。ユーザビリティの向上は、ユーザーがこれらの機能を活用するのに役立ちます。

リソース配分

モデル サービングは主にモデルのパラメーターを扱う必要があります。パラメーターの数とそれらの表現サイズは、必要なメモリ量を示しています。1 兆近くのパラメーターを扱う LLM でない限り、通常 GPU の一部しか必要しません。このことは、GPU のような高価なアクセラレーターを分割利用する必要性を浮き彫りにしています。まだアルファ版である DRA プロジェクト [53] は、GPU スケジューリングをより柔軟にすることを目指しています。

もう一つの考慮のポイントは応答レイテンシーで、これはユースケースによって大きく異なります。例えば、自律走行中に道路上の物体を検出するために必要な応答レイテンシーは、画像を作成したり詩を書いたりするときに許容できるレイテンシーよりも数段低いです。高負荷状態の低レイテンシー アプリケーションには、追加のサービング インスタンスを立ち上げる必要があるかもしれません。これらのインスタンスは、CPU、GPU、または望ましいレイテンシーが確保できるその他のコンピューティング リソースによって対策できます。Kubernetes では、利用可能なリソースに対するこのようなカスケード的なオポチュニスティック スケジューリングのサポートは、まだ発展途上です。

さらに、イベント駆動ホスティングは、リソースを無駄にせず、コストを抑えるのに理想的です。Kubernetes Event Driven Autoscaling(KEDA)[54] プロジェクトは、エンドツーエンドのサービス レイテンシーを実現するために、モデルの読み込みレイテンシーが許容可能であれば、このプロジェクトに適しています。共有に適した不変ファイル システムである Open Container Initiative (OCI)[55] フォーマットを使ったモデル共有のサポートの進化には新たなチャンスがあります。もう 1 つの解決策は、特に CN に AI を使用して、利用を予測し、予想される負荷に対処するために、サービング インスタンスをプロアクティブにフロートやシャットダウンすることです。

ユーザー エクスペリエンス

CN の特徴であるコンテナは持ち運びを可能にし Kubeflow のような Kubernetes の API とオペレーターは、AI ワークロードのデプロイを簡素化し、簡単にスケーラブルで「一度書けば（事実上）どこでも実行」できるようにします。ユーザーがベア メタルや仮想化環境上の従来のバッチ システムからコンテナや Kubernetes に移行すると、導入直後の利用時であっても、クラウド技術の利点を高く評価することができるでしょう。しかし、学習曲線は険しいかもしれません。

AI のトレーニング ワークロードについて考えてみましょう。ランタイム環境の設定は、特に高度にカスタマイズ可能なライブラリーを使用する場合、時間がかかることがあります。ユーザーは、多数の環境変数をデフォルト設定のまま使用することもできますが、これではパフォーマンスが悪い可能性があります。特定のトレーニング ワークロードを使って Kubernetes プラットフォームで最適化された場合には、別のプラットフォームやトレーニング タスク、または異なるライブラリが含まれるコンテナ バンドルで同じように実行される保証はありません。これは、ワークロードの移植性と使いやすさに影響します。

前項までで、データ準備、トレーニング、チューニング、サービング、ファインチューニングにまたがる、通常は多段階に及ぶ AI パイプラインの 1 段階だけを着目していました。システムやクラウドの概念に必ずしも精通していない AI 実務者にシームレスなユーザー エクスペリエンスを提供し、AI 開発の摩擦をなくす合理的な製品体験を提供するにはどうすればいいでしょうか。Kubernetes の複雑な詳細を簡略化することができる、ユーザーフレンドリーで、広く知られた Python による SDK を AI 実務者に提供することで、クラウド ネイティブ AI ツールの採用の増加に貢献することができます。ユーザーは、PyTorch や TensorFlow を使用して ML モデルを構築し、パッケージング、Docker イメージの構築、Kubernetes カスタム リソース（PyTorchJob、TFJob など）の作成、複雑なクラウド ネイティブ ツールを使用したモデルのスケーリングなどの詳細を気にすることなく、シンプルな Python SDK を使用して、迅速かつ簡単に Kubernetes インフラにデプロイしたいと考えています。よりユーザーフレンドリーな MLOps ライフサイクルのためのオープンソース製品エクスペリエンスを発明するためには、強力な製品開発に焦点を当てる必要があるでしょう。

JupyterLab のようにツールを統合することで、IDE のようなエクスペリエンスと、現在利用可能な AI/ML ツールに存在する有用な API（例：Kubeflow Katib API）を利用することができ、ML 実務者は複数のユーザー インターフェイスにまたがるコンテキストの切り替えを減らし、より迅速な AI 開発サイクルを回すことができます。JupyterLab の拡張可能な性質は、新しいツールやインターフェースを学ぶことなく、使い慣れたツールの中で AI/ML ワークロードを構築、デプロイ、監視するワークスペースを ML 実務者に提供します。Kubeflow Pipelines と組み合わせて Elyra[56] のような GUI ワークフロー構築ツールを使用して、個々の AI/ML ノートブックで開発されたコードのワークフローをスケジュールするために JupyterLab を使用することも可能です。

企業内外のビッグ データは AI の主役です。ビッグデータと ML のエコシステムのギャップをどのように埋めるかを検討することが不可欠です。例えば、最新の生成 AI モデルは、学習に大量のデータを必要とします。しかし、Iceberg のようなフォーマットから PyTorch のような学習フレームワークへ大量のデータをロードするためのツールは強化が必要であり、TorchArrow[57] や PyIceberg[58] のようなツールで初期の有効性が示されています。Spark のような大規模なデータ準備に使われるツールは、ML エコシステムのツールとうまく接続されていません。データを準備し、特徴量を構築し、特徴量をディスクに保存し、そしてトレーニング ワークロードで使用するためにそれらの特徴量をメモリに読み戻すには、余分なオーバーヘッドが必要となります。 RayData[59] のようなソリューションや、Arrow Flight RPC に基づいて構築されたデータ キャッシング マイクロサービスは、トレーニング ワークロードの最初のフェーズに関連する入出力のオーバーヘッドを大幅に改善する可能性があります。

ML ツールは複雑であり、Kubernetes にデプロイするには通常、ユーザーによるサポートが必要です。GPU の適切なドライバを特定してデプロイし、ユーザーの AI/ML ワークロードと互換性を持たせることは簡単ではありません。既存の ML ワークロードのアップグレードの手順は、他の Kubernetes のコントロール プレーン コンポーネントと同様に、簡素化や改善が必要です。ユーザーは、Kubernetes のアップグレードやクラスタのダウンタイムに対して、AI ワークロードの耐性を維持する方法についての明確なガイドラインも必要としています。

使いやすさに影響するもうひとつの側面は、クォーターやネームスペースを使ったマルチテナンシーです。管理者でないユーザーが、利用可能なシステム リソースを把握するためには助けが必要です。通常、管理者はオブザーバビリティー ツール（例えば、Grafana ダッシュボード）を提供しています。これらが欠けている場合、非熟練者 / 非管理者ユーザーは窮地に立たされます。

最後に、デバッグは困難であり、分散環境ではさらに難しく、処理パイプラインが複数の複雑なサービスで構成されている場合はなおさら難しくなります。ハードウェアやソフトウェアの障害は、クラウド ユーザーにとっては多かれ少なかれ明確で特定しやすいかもしれませんが、AI 実務者が障害の全体像を把握するには手助けが必要でしょう。例えば、NCCL の終了エラーは、多くの可能性のうち、どれが原因かが曖昧である場合があり、それぞれに調査が必要です。ユーザーはエラー メッセージを管理者に伝え、さらなる支援を求める必要があるかもしれません。

横断的な懸念

前のセクションでは、AI パイプラインの各ステージに特有の課題を取り上げました。しかし、リファレンス実装、オブザーバビリティー、セキュリティなど、すべてのステージとすべてのソフトウェア アプリケーションに共通する課題もあります。例えば、リソースの適切なサイジングは、データ処理、トレーニング、サービングに有効です。これはリソースの利用率、コスト、持続可能性に影響します。もう少し深く掘り下げてみましょう。

リファレンス実装

クラウドも AI も簡単な学問ではなく、多くのツールやプロジェクトの中から選択し、それら連携させることは簡単ではありません。採用においては概ねシンプルなユースケースと一致するリファレンス実装を要求することで、改善する必要があります。Kind for Kubernetes は、開発者がラップトップで作業を開始するのに大いに役立ちました。Jupyter Notebook は、新進の AI/ML 開発者にとっても同様でした。クラウド上で動作する AI/ML パイプラインにも同様のものが必要です。

リソース供給の適正化

AI/ML のワークロードは、特に数十億から数兆のパラメーターを持つ LLM では、リソースを大量に消費します。前述したように、GPU のようなアクセラレーターは高価で供給不足であり、リソースを節約しコストを抑制するためには、適切なサイズの割り当てで使用することが不可欠です。GPU をタイムスライスするだけでなく、分割セクションにスライスまたは区分し、異なるワークロードの要求に応じて適切に割り当てることができるようにする必要があります。上記のバックエンドの取り組みと同時に、GPU サブユニットを要求し、ワークロードを起動しながらそれらを設定するフロントエンドのサポートも必要です。

このニーズに対応するため、Kubernetes は v1.26 のアルファ版で新しい API、**Dynamic Resource Allocation (DRA),**[60] を導入しました [61]。この API は、特に特殊なハードウェア リソースを管理するための柔軟性を提供します。

- ネットワーク接続リソース
- リソース要求の任意パラメーター
- 任意のリソース固有のセットアップとクリーンアップ アクション
- リソース要求と利用可能なリソースとのカスタム マッチング

- DRA API には、既存のアプローチと比較していくつかの利点があります。
  - Kubernetes のコアとなるコードベースを変更することなく、DRA ドライバを開発してデプロイすることで、カスタム ハードウェアを追加できる
  - ベンダーはリソース パラメーターを定義できる
  - コンテナとポッド間でリソースを共有できる

コスト管理

AI/ML はすぐに予算のブラックホールになりかねません。AI クラウドのコストを最適化するためには、リソース割り当てとスケーリング プロセスの自動化が不可欠です。マイクロサービスは必要に応じて個別にスケーリングできます。さらに、Kubernetes の自動スケーリング機能を利用することで、アクティブ インスタンス数の適正化、ひいてはインフラ コストの適正化に役立ちます。最後に、**スポット インスタンス**は、サービス レベル アグリーメント（SLA）を満たしながらリスクをバランスさせるポリシーでの活用ができます。

オブザーバビリティー

オブザーバビリティーは AI/ML パイプライン全体で価値があります。CN は OpenTelemetry[62] や Prometheus[63] のようなツールを提供しており、負荷、アクセス数、応答レイテンシーなどを監視することができます。本番環境でモデルのパフォーマンスと健全性を監視することは極めて重要です。AI システムの精度と信頼性を確保するためには、モデルの偏差を追跡することが極めて重要です。例えば、顔認識システムは、COVID-19 のパンデミック時にマスクを着用する人が増えることにより性能低下する可能性がありました。同様に、住宅価格予測モデルは、自然災害や金利の変化などの外部要因によって現実から乖離する可能性があります。したがって、AI モデルを継続的にモニタリングすることは、パフォーマンスの問題を検出し、必要な調整を行うために不可欠です。

インフラ監視は、特に長時間稼働するワークロードでは不可欠です。AI トレーニングのワークロードが実行されると、GPU やネットワークに異常が発生することがあります。例えば、GPU メモリのエラーや到達不能なノードがあり、その結果ジョブがクラッシュすることがあります。

また、すぐに発生しない問題が起きる場合もあります。例えば、明らかなハードウェア障害が報告されないまま、トレーニングのパフォーマンスが低下し始めるような場合です。このような場合、深層診断だけが問題を特定できます。現在のメトリクスでは、深層診断の結果は公開されていません。したがって、AI トレーニング ジョブの実行前、実行中、実行後にインフラの問題を検出、回避、対処するツールを提供することが重要になります。

災害復旧と事業継続

すべてのプロダクション サービスは、バックアップによって柔軟性を持たなければなりません。AI サービスも同様です。サービスに障害が発生したり、対応が遅れたりすると、風評被害や収益の損失を招く可能性があります。データのバックアップ、複数のアベイラビリティ ゾーンでのインスタンスの実行、複数のインスタンスの実行など、包括的な災害復旧計画の策定が不可欠です。ポリシーはこれらを支援することができます。

セキュリティおよびコンプライアンス監査

すべての外部向けサービス、特にモデル サービング インスタンスには、ファイアウォール保護、アクセス制御などが必要です。また、他のサービスと同様に、AI/ML ワークロードはセキュリティのベストプラクティスに従わなければなりません。これには、侵入テスト、脆弱性スキャン、医療、金融などのワークロード領域のコンプライアンス チェックなどが含まれます。

Grype[64] や Trivy[65] のようなツールは、コンテナ化されたワークロードの脆弱性をスキャンすることができます。Kyverno[66] やポリシー実施サービスは、コンテナ化されたワークロードが必要最小限の機能で必要最小限の権限で実行されていることを確認することができます。

機密コンピューティング [67] や信頼された実行環境（TEE）を使用することで、さらに一層のセキュリティ対策が可能です。これらのハードウェアがサポートする環境は、暗号化されたメモリ、データ整合性保護、およびテスタビリティを提供します。TEE は、使用中のデータとワークロードを他のインフラ ユーザーから保護します。AMD、Intel、NVIDIA、IBM は TEE を提供しており、パブリック クラウドでも利用できるようになってきています。医療や金融情報、ML モデルなどの機密データの保護は、主要なユースケースです。

持続可能性

AI/ML モデルのトレーニングは、特に GPT-3 のような大規模言語モデルでは、常にリソースを消費してきました。学習による排出量は複数回の大陸横断フライトに匹敵し、推論による排出量は大量のクエリにより、さらに増加します [68]。市場優位性を求めてモデルを大型化する業界の傾向は非効率を招き、エネルギーと資源の消費を助長しています [69]。モデルの環境影響を報告する際の透明性や標準化の改善が課題です。

最近では、ChatGPT のような LLM を稼動させているサーバーを冷却するための水の使用量に関するいくつかの見識が示される始めている一方で、LLama を用いた透明性の改善の取り組みも行われています [70]。ChatGPT の二酸化炭素排出量は、数百万人のユーザーを考えると、重要です。

持続可能性の推進は、イノベーションの機会をもたらします。DeepMind の BCOOLER や、DistilBERT や FlexGen のようなより小さい効率的なモデルでは AI/ML によるエネルギー消費の削減を約束しています [71]。効率的な ML アーキテクチャ、最適化されたプロセッサ、エネルギー効率の高い場所へのクラウド コンピューティング インフラの設置といったベストプラクティスを採用することで、ML トレーニングの二酸化炭素排出量を抑制することができます。グーグルは、機械学習システムのエネルギー消費の抑制に成功しています。

子供のための教育

今日、技術教育は主に、AI やコンピュータによる支援を伴わない従来のプログラミング言語に焦点を当てています。学校は通常、リファクタリング、テンプレート化、API アシストをサポートする最新の IDE を使用せず、セットアップを容易にするために、学生に自前のウェブサイト上でコーディングさせています。また、将来の標準的な開発モードになるにもかかわらず、Github の Copilot のような AI のコーディング支援技術の使用も教えていません。ほとんどの学生は、この技術が存在することすら知らないのです。

学校は、不正行為への懸念から、ChatGPT や Copilot のような AI 技術を生徒に積極的に使わせないようにしています。そのため、生徒たちは AI 技術を使って仕事を補強したり、効果的に活用する方法を学ぶことができません。学校は AI 技術に否定的なイメージを与えているため、勉強熱心な生徒は AI を使うことを怖がり、宿題をやらずに済む方法を探している生徒が AI を使う可能性が高くなります。

上記の課題は、CNAI システムを導入する際に懸念される事象について示しています。幸いなことに、CN ツールは多くの課題に正面から向き合っています。次に、これらの課題から派生する新たなチャンスについて考察します。

クラウド ネイティブ人工知能で前進する道

本セクションでは、CNAI を率先して導入するための前向きなアプローチを提供します。まず推奨事項（または行動）から説明し、次に既存の、まだ発展途上のソリューション（すなわち CNAI ソフトウェア）を列挙し、最後にさらなる発展の機会を検討します。

推奨事項

柔軟性

AI に関するさまざまな選択肢に圧倒されることがあります。幸いなことに、多くの人々のおかげで、人気のツールや技術はこの新しい世界でも有用であり続けています。REST インターフェイスからクラウド ベースのリソースやサービスまで、CN 技術は現在も活用できる状態であり、新しいサービスが進化しても活用し続けることができるでしょう。

持続可能性

AI ワークロードが環境に与える影響の説明性を向上させることは、特にクラウド ネイティブの世界において、生態系の持続可能性の維持に極めて重要です。これは、生態系の持続可能性に関する AI ワークロードの明確化、分類、触媒を支援するプロジェクト、方法論、分類法をサポートすることで達成できます。さらに、AI ワークロードのスケジューリング、オートスケール、チューニングを最適化するためにクラウド ネイティブ技術を統合することも必要です。さらに、環境影響評価における標準化された方法論の採用を提唱することが不可欠です。また、エネルギー効率の高い AI モデルの開発と利用を促進し、主に Kubeflow のようなクラウド ネイティブ スタックを通じて、モデルの開発と利用における透明性を促進することも重要である。最後に、目的を持った効率的な AI 利用の重要性を強調することで、不必要な計算負荷を最小限に抑えることができます。

カスタム プラットフォームの依存関係

クラウド ネイティブ環境に必要な GPU ドライバが準備されており、AI ワークロードのための GPU アクセラレーションをサポートしていることを確認することをお勧めします。AI アプリケーションは特定のフレームワークやライブラリ バージョンに依存することが多く、標準的なコンテナ イメージに簡単にアクセスできなかったり、互換性がなかったりする場合があるため、これは非常に重要です。これは、さまざまなベンダーや GPU アーキテクチャを持つという課題を解決するのに役立ちます。

リファレンス実装

AI 開発に関わるツールの数と複雑さを考えると、クラウドで AI/ML を素早く始めるために、世界中のどのチームにも製品のような体験を提供できる様々なツールが使いやすく組み合わされているクラウド ネイティブである、OpenTofu ベースのリファレンス実装の価値を検討することが望ましいかもしれません。データ準備、特徴量ストア、トレーニング、チューニング、モデル登録、およびサービングに利用可能な最高のオープンソース ツールを組み合わせることで、チームは機械学習を迅速に開始し、クラウドのパワーを使用して作業を効率的にスケール アップすることができます。このような目的のために、洗練された一連の技術を機能的でスケーラブルなディストリビューションに組み合わせることの価値とパワーを考えてみましょう。（例えば、JupyterLab、Kubeflow、PyTorch、Spark/Ray/Trino、Iceberg、Feast、MLFlow、Yunikorn、EKS/GKE、S3/GCS など）。このようなリファレンス実装は、クラウド ベースの技術によるオープンで責任ある AI/ML 開発を推進する上で、非常に価値があると思われます。

業界用語の受け入れ

AI がユビキタスになるにつれて、ある側面ではますます複雑になる一方で、ある側面ではよりシンプルになっています。例えば、専門用語が進化し、AI についてより簡単に会話できるようになっています（例えば、既存のコンテンツを再利用することを意味する「repurpose」のような用語）。これは、RAG、Reason、Refinement といった、より専門的な用語にも当てはまります。

進化する AI/ML ソリューション

以下は、CNAI を含む AI を実現するための選択肢となった具体的なツールや技術のほんの一例です。

オーケストレーション - Kubeflow

Kubeflow は、ML オペレーション（MLOps）をサポートする CNAI ツールの一例です。Kubernetes、ステートレス アーキテクチャ、分散システムなどの技術を使用して、Kubeflow は AI/ML コミュニティーがクラウド ネイティブ ツールの導入を効率化しています。Kubeflow の成功した採用事例は、AI/ML/DL のためのクラウド ネイティブ技術の統合の成功を強調しています。Kubeflow は、Kubernetes が提供する伸縮性のある基盤に機械学習のコンセプトを適用する能力において非常に先進的であり、他の多くのプロジェクトもこれに追随しています[72]。Kubeflow は Kubernetes のベストプラクティスに従い、AI/ML の領域に宣言型 API、コンポーザビリティ、移植性などを適用しています。Kubeflow は、ML ライフサイクルの各段階に個別のマイクロサービスを実装しています。例えば、Kubeflow Training Operator は分散型トレーニングに、Katib はハイパーパラメーター チューニングの微調整に、Kubeflow KServe はモデル サービングに使用されています。これにより、ユーザーは個々の Kubeflow コンポーネントを ML インフラに統合したり、エンドツーエンドの ML プラットフォームとして Kubeflow を利用したりすることができます。

コンテキスト – ベクトル データベース

LLM は、ある時点で、一般に公開されている膨大な量のデータを使って訓練されます。私たちはプロンプトを通じて LLM と対話しますが、その際、ユーザーが長いプロンプトや複数のプロンプトを入力することなく、より価値のある回答やドメインに特価した回答を得るためには、プロンプトを「充実」させることが有用です。そこでベクトル データベースの出番です。ベクトル データベースは、ベクトル（数値で表現されたデータの数学的表現）のインデックスがつけられた巨大なストレージです。エンベディングは、追加された各データの特定のベクトル表現です。多くの場合、独自の、ドメイン固有の、または、より新しいもので、データ間の関係や類似性（コンテキスト）を表現することを目的としています。ユーザーから提供された LLM プロンプトは、ベクトル データベースで使用されているのと同じエンベディングを使用して変換され、得られたベクトルはデータベース内の類似したベクトルを見つけるために使用されます。そして、LLM に入力して応答を生成する前に、追加のコンテキストを提供するためにマージされます。マルチモーダル GenAI システムは、テキスト、画像、音声、またはその他のプロンプトを扱うことができ、多様な入力を処理する埋め込み能力を持ちます。

ベクトル データベースは、専用に構築されたものもあれば、従来のデータベースをベクトルをより特別に扱うために拡張したものもあります。インデックス作成方式、類似度を計算するための距離メトリクス、データ圧縮技術の有無など、その種類はさまざまです。Redis[73]、Milvus[74]、Faiss[75]、Weaviate[76] などの種類があります。

オブザーバビリティー - OpenLLMetry

OpenLLMetry[77] は OpenTelemetry[78] の上に構築されたプロジェクトで、LLM オブザーバビリティーのための徹底してベンダーに依存しない計測を可能にします。生成 AI は従来の感覚でのデバッグができない（つまり、”コードをステップ スルーする”ことができない）ため、開発者は生成 AI の使用を長期的に改善するために、オブザーバビリティー ツールやプラクティスに目を向けなければなりません。このデータはまた、しばしば評価やワークフローの微調整のために使用されます。

今後のチャンス

CNCF プロジェクトの状況

CNCF、LF AI[79] & Data、AI Alliance などのパートナー [80] など、いくつかの Linux Foundation(LF) グループは、AI エンジニアとクラウド エンジニアの両方が利用できる AI プロジェクトのハブを提供しています。クラウド ネイティブ ランドスケープ（Cloud Native Landscape）[81] などの既存のツールは、CN エコシステムを俯瞰することができます。次の図は、確立されたプロジェクトと発展中のプロジェクトを機能分野別に分類したものです。

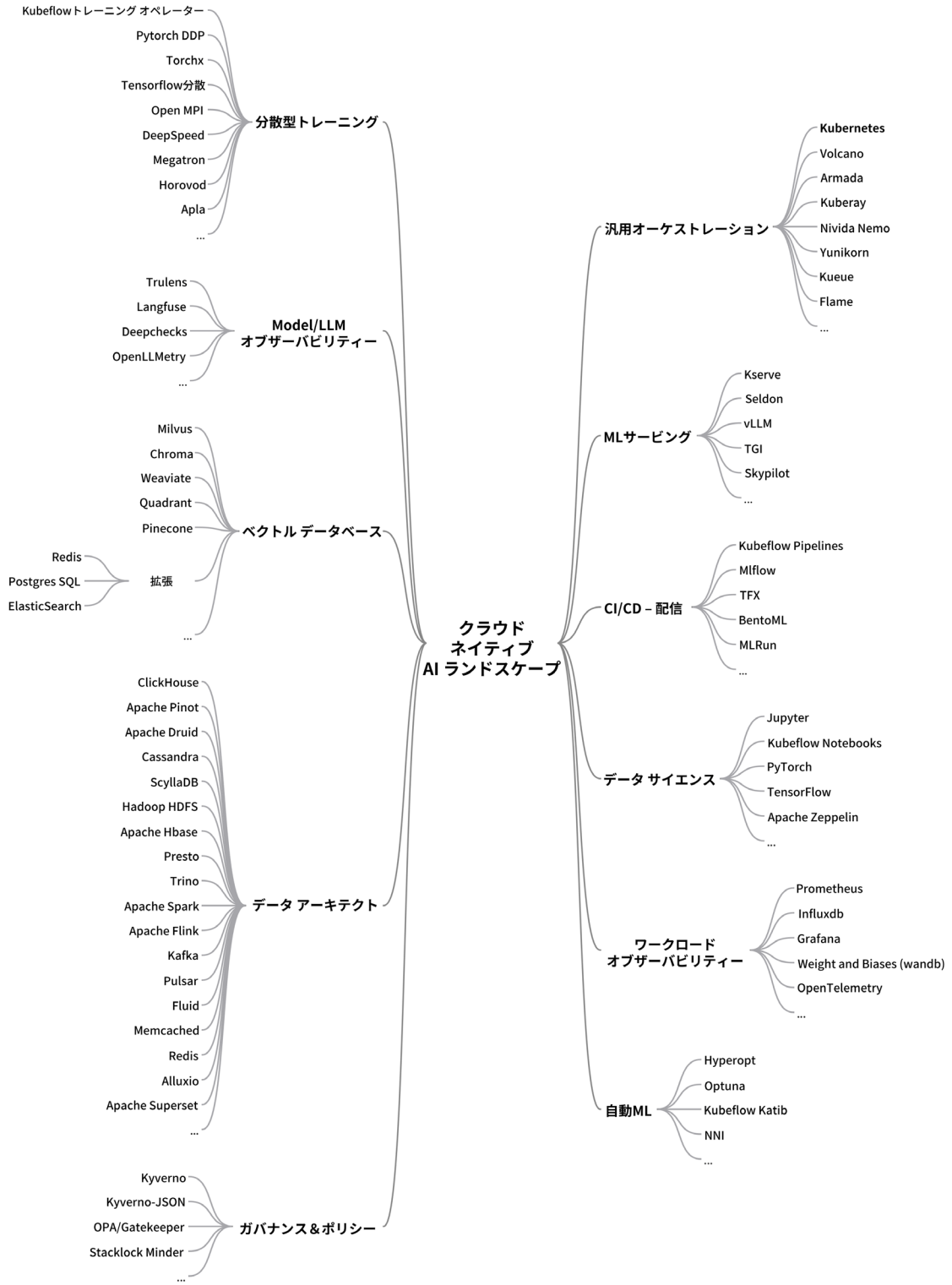


図 4
ML ツールからタスクへのマインド マップ

子供と学生のための CNAI

子供たちはすでに ChatGPT のような AI 支援技術を毎日使っていますが、それがどのように機能するのか知りません。識別 AI アルゴリズムや生成 AI アルゴリズムのような現代 AI の基盤は、子供たちやテクノロジーに精通した親でさえ理解できないブラックボックスであり、興味を持つことは難しいです。ChatGPT のような LLM をただ鵜呑みにするのではなく、学生の教育にはニューラル ネットワークや機械学習アルゴリズムの基礎を盛り込み、AI 技術がどのように機能し、将来のキャリアでどのように活用できるかを説明する必要があります。

クラウド ネイティブ コミュニティーや、KubeCon での CNCF Kids Day[82] のような成功事例では、クラウド ネイティブや AI 技術に関する教育の機会を提供しています。子供たちに早くから AI 技術を紹介することは、コンピュータ サイエンスを悩ませている多様性、公平性、インクルージョンの問題を防ぐことにもなります。あらゆる人種、性的指向、社会経済的地位の人々が AI/ML を日々体験し、適切な訓練と教育によってこの技術の改善に貢献することができるため AI は平等化に貢献する技術であるともいえます。

AI/ML 革命は、ウェブ技術がユビキタスになり、一般労働者までもがビジネスを改善するためにこの技術を取り入れたドットコム時代に似ています。AI/ML 技術が社会にユビキタスになるにつれ、学生たちが AI やクラウド ネイティブ技術の進歩に遅れないようにしなければなりません。

参加

AI が成長するにつれ、教育やかかわりあう機会が増えます。AI スペシャリスト（ML の博士号取得者やデータ サイエンティストなど）や AI のジェネラリスト（オペレーターやエンドユーザーなど）の活躍の場があります。MOOCs[83] や認定資格のような教育プログラムは、あらゆる面で AI のツールや技術に焦点を当てるために出現しました。専門家の協会（ACM[84] や IEEE[85] など）やミートアップは、実際に会って学び、課題について議論する機会を提供しています。CNCF[86]、Linux Foundation AI*、AI Alliance[87] などの業界団体は、プロジェクトやプロトコルを大規模に調整する能力を提供しています。

信頼と安全 / セーフティ バイ デザイン

AI やクラウド ネイティブ技術を構築する際、意図しない結果や悪影響を及ぼす重大なリスクがあります。これらは、例えば、憎悪に基づく暴力的で過激な題材を不用意に宣伝するアルゴリズムを推奨するなど、意図しない設計上の問題によって、社会的弱者に悪影響を及ぼす可能性があります。また、生成 AI ツールを使って誤報や偽情報キャンペーンを行ったり、LLM を意図的に微調整して児童性的虐待の資料を作成したりするなど、個人やグループが意図的にシステムやツールを悪意を持って使用し、故意に危害を加えることもあります [88]。

AI とクラウド ネイティブ技術は、Trust and Safety、即ち「コンテンツを管理し、ユーザーや他者に対するリスクを把握し、オンラインなどの技術による虐待を緩和し、ユーザーの権利を守り、ブランドの安全を守るための、デジタル サービスが採用する分野と実践」[89] で使用されるツールの中核です。暴力的な可能性のあるふるまいの特定と評価、案件の選別と優先順位付け、強制執行の決定と記録、介入策の選択と適用、脅威情報の収集など、Trust and Safety サイクル [90] のあらゆる部分を実現するために、システムが構築されています。これらのシステムは、インターネットの安全性や健全性の中核であることとは別に、十分な配慮なしに設計された場合、重大な悪影響を及ぼす可能性があります。

責任ある技術とは、技術による危害を減らし、技術パイプラインを多様化し、技術が公共の利益に合致するようにすることです。リスクと危害を管理し軽減するためには、技術の価値、意図しない結果、悪影響を探り、積極的に考慮する必要があります。AI やクラウド ネイティブ技術を構築する際、こうした潜在的な倫理的・人権的影響を考慮し、

* 訳注：この ”Linux Foundation AI” は、”Linux Foundation” または ”LF AI & DATA” と思われる。

表現の自由、プライバシーの権利、生命、自由、人の安全に対する権利[91]、その他の基本的普遍的人権を最適化しなければなりません。

世界経済フォーラムでは「セーフティ バイ デザインは、ユーザーの安全と権利を、オンライン製品やサービスの設計と開発の中心に据えるものである。」[92] と説明されています。この積極的で予防的なアプローチは、安全を組織の文化やリーダーシップに根付かせることに重点をおいています。説明性を重視し、すべての人にとってより前向きで、市民的で、やりがいのあるオンライン体験を促進することを目的としています。

テロ対策に関するグローバル インターネット フォーラム（GIFCT）[93]、Tech Coalition[94]、the Internet Society[95] などの開発のベストプラクティスを支援する専門家の分野も広がっています[96]。AI Alliance[97] の取り組み（IBM、Meta など 50 以上の機関）は、閉鎖的な AI システムに代わるものを提案し、責任ある AI（倫理、信頼、安全）の分野を前進させるため、AI におけるオープン イノベーションと科学の推進に焦点を当てています。ChatGPT を支える組織である OpenAI[98] は当初、AI における安全性と公平性の保証に焦点を当てた非営利団体として設立されました。

新しい工学分野の出現

過去 20 年間、私たちは、技術業界がエンジニアの職責に応じて、急速にエンジニアの職能を創出し、変化させてきた様子を見てきました。私たちは、DevOps エンジニア、SRE エンジニア、インフラ エンジニアといった役割の台頭を目の当たりにしてきました。私たちは、MLDevOps や AI エンジニアが、今後数カ月から数年のうちに、データ サイエンス、インフラ、開発の間のつなぎ役になることを予見しています。この業界領域は発展途上であり、役割の名前は変化する可能性があることを知っておくことが重要でしょう。また、異なる用語が現れるかもしれません。将来的には、その役割は、AI ツール、インフラ、AI チェーンとエージェントの展開により重点を置くことになっていくでしょう。

クラウド ネイティブのための人工知能

本稿では主に、AI の開発と利用をサポートするクラウド ネイティブに焦点を当ててきました。しかし、AI は様々な方法でクラウド ネイティブを強化することができます。特に、省電力、リソース利用率の向上、待ち時間の短縮、優先順位の遵守、セキュリティの強化、ログやトレースの理解など、複数の最適化基準が関係します。

クラスタ制御のための自然言語インターフェース

2023 年にシカゴで開催された Cloud Native AI + HPC Day[99] では、自然言語インターフェースを備えた Kubernetes Controllers がクラスタ関連のタスクに取り組むデモが行われました。そのバックエンドでは LLM が使用され、ユーザーリクエストを理解し、Kubernetes API 呼び出しに変換していました。さらに、サービスの回復力を確認したり、CVE をスキャンしたりするためのカオス テストの開始もサポートしました。これは、Kubernetes クラスターのより直感的なオーケストレーションと管理への先駆けであり、やがて管理者やサイトの信頼性エンジニアの学習を容易にする取り組みです。

セキュリティ

機械学習は、膨大なデータセットを分析してパターンを迅速に特定し、システムの潜在的な脅威や弱点を予測することができます。レッド チーム [100] に AI を組み込むことで、セキュリティ課題の特定を加速し、組織は新たなサイバー脅威に対する防御を強化することができます。異常なネットワーク動作を検出する ML モデルは、ワークロードを保護するためにクラスタ内でも、エッジ展開のためのクラスタ群全体でも、同じように簡単に使用することができます。

よりスマートなオーケストレーション / スケジューリング

AI は、1 日 /1 週間 /1 か月のクラスタ使用履歴を分析し、ワークロードのパターンとリソースの可用性を特定することで、ワークロードをいつ、どのように展開するか、水平または垂直にスケールさせるか、いつワークロードを少数のノードに集約し、他のノードを休止状態にして電力を節約するか、あるいはクラスタから削除してコストを削減するかを理解することができます。

ML 駆動モデルは、タスクの順序付けを最適化し、意思決定プロセスを自動化し、ワークロード管理の全体的な効率を高めることができます。自然言語インターフェースは、オーケストレーションとスケジューリングのプロセス全体を簡単化します。このような機能強化により、組織は動的なクラウド環境における複雑なワークフローの管理とスケジューリングが容易になります。プロセッサーの消費電力モデルは、消費電力削減のための計画と最適化を支援するために構築されています。

取り組み中および探査中の AI 統合の取り組み

- カスタム LLM を微調整してログを分析
- MLOps パイプラインでデータの出所を把握し、メンテンナンスする
- OpenTelemetry[101] のような CNCF プロジェクトでの AI の意味的規約
- AI を搭載した開発環境（IDE）の AI アプリケーション開発・デプロイへの使用

この分野での進歩については、そう遠くない将来に報告することができるでしょう。

結論

人工知能（AI）とクラウド ネイティブ（CN）技術の組み合わせは、企業が前例のない能力を開発する絶好の機会を提供しています。クラウド ネイティブ インフラの拡張性、回復力、使いやすさを利用すれば、AI モデルをより効率的に、より大規模にトレーニングしたりデプロイしたりすることができます。このホワイトペーパーでは、この 2 つの領域の交点を深堀りし、組織がこの強力な組み合わせを活用するための現状、課題、機会、潜在的なソリューションについて議論してきました。

複雑な AI ワークロードのリソース需要の管理、AI モデルの再現性と解釈可能性の確保、技術者でない実務者のユーザー エクスペリエンスの簡素化など、いくつかの課題が残る一方で、クラウド ネイティブ エコシステムはこれらの懸念に対処するために継続的に進化しています。Kubeflow、Ray、KubeRay のようなプロジェクトは、クラウドで AI ワークロードを実行するための、より統一されたユーザーフレンドリーな体験への道を開きます。さらに、GPU スケジューリング、ベクトル データベース、持続可能性に関する現在進行中の研究は、制限を克服するための有望なソリューションを提供するでしょう。

AI とクラウド ネイティブ技術が成熟するにつれ、このシナジーを取り入れる組織は、大きな競争優位性を解き放つことでできるでしょう。複雑なタスクの自動化や膨大なデータセットの分析から、クリエイティブなコンテンツの生成やユーザー エクスペリエンスのパーソナライズまで、その可能性は無限大です。適切な人材、ツール、インフラに投資することで、組織は AI とクラウド ネイティブ技術の力を活用し、イノベーションを推進し、オペレーションを最適化し、卓越した顧客体験を提供することができるでしょう。

本稿は **CNCF AI ワーキング グループ**が作成

付録

参考文献

1. 用語集
2. 参考文献

用語集

AI 実務者

本稿では、ML エンジニア、データ サイエンティスト、データ エンジニアを指しています（ただし、これらに限定しているわけではありません）。データ エンジニアは、関連データの操作、機械学習モデルの作成、最適化を主な職務とします。

開発者

本稿では、ソフトウェア エンジニア、フロントエンド エンジニア、バックエンド エンジニア、フル スタック エンジニア、ソフトウェア アーキテクト、ソフトウェア テスターを指しています（ただし、これらに限定しているわけではありません）。ユーザー インターフェース、マイクロサービス、バックエンド ソフトウェアを含むソフトウェアの作成とテストを主な職務とします。

デプロイヤー

本稿では、DevOps エンジニア、サイト信頼性エンジニア、インフラ エンジニア、インフラ アーキテクト、アプリケーション管理者、クラスタ管理者を指しています（ただし、これらに限定しているわけではありません）。開発時、ステージング時、本番を含む複数の環境へのソフトウェアとクラウド インフラのデプロイを主な職務とします。

DRA

DRA は Dynamic Resource Allocation の略。一般的なリソースの要求とポッドの提供を API で抽象化したもので、Kubernetes のコア API を書き換えることなく、サード パーティー ベンダーがオンデマンドで HW/SW リソースを提供できるようにします。

LLM

LLM とは大規模言語モデル（Large Language Model）の略。大規模言語モデルとは、人間のようなテキストを理解し生成するために、膨大な量のテキスト データで学習された人工知能モデルです。LLM は、自然言語処理（NLP）タスクのために特別に設計された機械学習モデルのサブセットです。

LLMOps

LLMOps とは、Large Language Model Operations の略で、大規模言語モデル（LLM）に特化した運用面全般を意味します。本質的には、LLMOps は LLM を利用したアプリケーション特有の要件に対し、MLOps の原則とツールを適応させたものであり、開発からデプロイ、メンテナンスまでのライフサイクル全体を含んでいます。

MIG

Multi-Instance GPU 技術の略で、1 つの物理 GPU（グラフィックス プロセッシング ユニット）を複数のインスタンスに分割し、それぞれが独自のリソースと機能を持つ独立した GPU として動作することを可能にする革新的な技術です。この技術により、データ センターおよびクラウド コンピューティング環境における GPU の利用率と柔軟性が向上します。

MLOps

MLOps とは、機械学習オペレーションの略で、本番環境における機械学習モデルのデプロイ、モニタリング、管理を合理化したり自動化したりするために使用されるプラクティス、方法論、ツールを指します。MLOps は、機械学習の開発と運用のギャップを埋めることを目的とし、ML モデルが効率的で、信頼性が高く、大規模にデプロイされることを保証します。ソフトウェア エンジニアリングの原則、DevOps プラクティス、専用ツールを組み合わせて、データの準備、モデル トレーニング、モデルのデプロイ、モニタリング、メンテナンスなど、エンドツーエンドの ML ライフサイクルを自動化します。MLOps は、組織が ML プロジェクトを加速し、モデルのパフォーマンスを向上させ、ML パイプライン全体の一貫性と信頼性を維持するのに役立ちます。

MPS

MPS とは、GPU コンピューティングにおけるマルチプロセス サービスの略です。MPS 技術は、分離と効率的なリソース利用を維持しながら、GPU によって高速化する複数のアプリケーションまたはプロセスが単一の物理 GPU を共有することを可能にします。

RAG

AI において、RAG は”Retrieval-Augmented Generation”の略です。これは、テキストを生成するために検索ベースモデルと生成モデルを組み合わせたモデル アーキテクチャです。RAG の生成プロセスは、モデルが広範なデータベースや知識ベースから関連情報にアクセスするのを助ける検索メカニズムで補強されています。この検索コンポーネントにより、モデルは外部の知識を生成プロセスに組み込むことができ、生成されるテキストの品質と関連性が向上します。

vGPU

vGPU(Virtual Graphics Processing Unit) 技術は、複数の仮想マシン（VM）が単一の物理 GPU(Graphics Processing Unit) を共有することを可能にします。この技術は、クラウド コンピューティング、データ センター、仮想デスクトップ インフラ（VDI）などの仮想環境で GPU リソースを効率的に利用します。

本ドキュメントは、CLOUD NATIVE ARTIFICIAL INTELLIGENCE の参考訳です。
翻訳協力：辻村幸弘

参考文献

1 https://github.com/cncf/toc/blob/main/DEFINITION.md

2 https://en.wikipedia.org/wiki/Microservices

3 https://landscape.cncf.io/guide

4 https://docs.aws.amazon.com/whitepapers/latest/build-secure-enterprise-ml-platform/personas-for-an-ml-platform.html

5 First release of Docker March 20, 2013.

6 https://en.wikipedia.org/wiki/LXC

7 https://en.wikipedia.org/wiki/Docker_(software)

8 https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/44843.pdf

9 https://github.com/cncf/toc/blob/main/DEFINITION.md as of Jul 18, 202

10 https://en.wikipedia.org/wiki/DevOps

11 https://about.gitlab.com/topics/gitops/

12 https://ai100.stanford.edu/2016-report/appendix-i-short-history-ai

13 https://youtu.be/P18EdAKuC1U?si=Dd74AdpbF3EgzVmn

14 https://www.cs.toronto.edu/~hinton/absps/NatureDeepReview.pdf

15 https://arxiv.org/abs/2008.02217

16 https://openai.com/chatgpt

17 https://en.wikipedia.org/wiki/Prompt_engineering#Retrieval-augmented_generation

18 https://github.com/zanetworker/ai-landscape

19 https://openai.com/research/scaling-kubernetes-to-7500-nodes

20 https://huggingface.co/blog/hugging-face-endpoints-on-azure

21 https://kubernetes.io/docs/concepts/scheduling-eviction/kube-scheduler/

22 https://github.com/intel/platform-aware-scheduling/tree/master/gpu-aware-scheduling

23 https://kubernetes.io/docs/tasks/manage-gpus/scheduling-gpus/

24 https://opencontainers.org/

25 https://k8sgpt.ai/

26 https://docs.aws.amazon.com/whitepapers/latest/build-secure-enterprise-ml-platform/personas-for-an-ml-platform.html

27 https://docs.databricks.com/en/machine-learning/mlops/mlops-workflow.html

28 https://www.ibm.com/topics/machine-learning-pipeline

29 https://cloud-native.slack.com/archives/C05TYJE81SR

30 https://www.intel.com/content/www/us/en/newsroom/resources/moores-law.html

31 https://gdpr-info.eu/

32 https://oag.ca.gov/privacy/ccpa

33 https://iapp.org/news/a/5-things-to-know-about-ai-model-cards/

34 https://arxiv.org/abs/2205.07147

35 https://kubernetes.io/docs/concepts/scheduling-eviction/dynamic-resource-allocation/

36 Open Source for Sustainability

37 https://github.com/kserve/kserve/

38 [2112.06905] GLaM: Efficient Scaling of Language Models with Mixture-of-Experts

39 A carbon-aware workload dispatcher in multi-cluster Kubernetes environments for Cloud

Native Sustainability Week 2023 | IBM Research

40 https://mlco2.github.io/impact/

41 https://codecarbon.io/

42 https://yunikorn.apache.org/

43 https://volcano.sh/

44 https://kueue.sigs.k8s.io/

45 https://en.wikipedia.org/wiki/Gang_scheduling

46 https://arxiv.org/abs/1706.03762

47 https://www.samsara.com/blog/building-a-modern-machine-learning-platform-with-ray

48 https://cloud.google.com/blog/products/ai-machine-learning/build-a-ml-platform-with-kubeflow-and-ray-on-gke

49 https://kueue.sigs.k8s.io/docs/tasks/run_rayjobs/

50 https://cloud.google.com/blog/products/containers-kubernetes/use-ray-on-kubernetes-with-kuberay

51 https://www.redhat.com/en/blog/fine-tuning-and-serving-open-source-foundation-model-red-hat-openshift-ai

52 https://kubernetes.io/docs/concepts/scheduling-eviction/assign-pod-node/

53 https://kubernetes.io/docs/concepts/scheduling-eviction/dynamic-resource-allocation/

54 https://keda.sh/

55 https://en.wikipedia.org/wiki/Open_Container_Initiative

56 https://github.com/elyra-ai/elyra

57 https://pytorch.org/torcharrow/beta/index.html

58 https://py.iceberg.apache.org/

59 https://docs.ray.io/en/latest/data/data.html

60 https://kubernetes.io/docs/concepts/scheduling-eviction/dynamic-resource-allocation/

61 https://github.com/kubernetes/enhancements/tree/master/keps/sig-node/3063-dynamic-resource-allocation

62 https://opentelemetry.io/

63 https://www.cncf.io/projects/prometheus/

64 https://github.com/anchore/grype

65 https://github.com/aquasecurity/trivy

66 https://github.com/kyverno/kyverno

67 https://en.wikipedia.org/wiki/Confidential_computing

68 https://www.cutter.com/article/large-language-models-whats-environmental-impact

69 https://marksaroufim.substack.com/p/moral-language-models

70 https://arxiv.org/pdf/2302.13971.pdf

71 https://analyticsindiamag.com/the-environmental-impact-of-llms/

72 https://landscape.lfai.foundation/

73 Redis

74 Vector database - Milvus

75 facebookresearch/faiss: A library for efficient similarity search and clustering of dense vectors. (github.com)

76 _

77 https://github.com/traceloop/openllmetry

78 https://opentelemetry.io/

79 https://lfaidata.foundation/

80 https://thealliance.ai/

81 https://landscape.cncf.io/

82 https://events.linuxfoundation.org/kubecon-cloudnativecon-europe/program/kids-day/#kids-day

83 https://www.mooc.org/

84 https://www.acm.org/

85 https://www.ieee.org/

86 https://www.cncf.io/

87 https://thealliance.ai/

88 CSAM - child sexual abuse material

89 https://dtspartnership.org/wp-content/uploads/2023/01/DTSP_Trust-Safety-Glossary13023.pdf

90 https://docs.google.com/document/d/1sXo-T3oEGcRTWmJJ_PUWwWyrvELi-mcEB3_27Nj_xkM/

91 https://www.ohchr.org/documents/publications/guidingprinciplesbusinesshr_en.pdf

92 https://www.weforum.org/projects/safety-by-design-sbd/

93 https://www.gifct.org

94 https://www.technologycoalition.org/

95 https://www.internetsociety.org/

96 https://alltechishuman.org/responsible-tech-organizations

97 https://thealliance.ai/news

98 https://openai.com/about

99 https://www.youtube.com/watch?v=1oTx7kgGeMg

100 https://en.wikipedia.org/wiki/Red_team

101 https://github.com/open-telemetry/semantic-conventions/issues/327

THANK YOU!

CLOUD NATIVE
COMPUTING FOUNDATION