インテル® AI for Enterprise RAG を使用したマルチノード・デプロイメント

この記事は、インテルのブログで公開されている「Multi-node deployments using Intel® AI for Enterprise RAG」の日本語参考訳です。原文は更新される可能性があります。原文と翻訳文の内容が異なる場合は原文を優先してください。

独自データから知見を引き出すため生成 AI を導入する企業が増えるにつれ、スケーラブルで効率的、かつハードウェア対応のインフラストラクチャーの必要性が極めて重要になっています。インテル® AI for Enterprise RAG (英語) は、**検索拡張生成 (RAG: Retrieval-Augmented Generation)** 向けのモジュール式で実稼働環境対応のフレームワークを提供することで、この課題に対処します。RAG は、エンタープライズ・データソースから取得したドメイン固有の知識を用いて LLM レスポンスを強化する手法です。

最新の Kubernetes クラスター上で動作するように構築され、**インテル® Xeon® プラットフォーム**と**インテル® Gaudi® AI アクセラレーター**向けに最適化されたインテル® AI for Enterprise RAG は、高度なポッド・スケジュール、リソース分離、動的スケーリングを活用して、多様な環境で推論を提供します。この記事では、このソリューションを複数のノードにスケールするため、ハードウェア・トポロジーをインテリジェントに検出し、予測可能で効率的な AI ワークロード向けにリソースを分離する**ノード・リソース・インターフェイス (NRI) プラグイン** (英語) を活用する仕組みについて説明します。

複数ノードにソリューションをスケールする

マルチノードの Kubernetes 環境にソリューションを展開する際の最初の課題の 1 つは、特に顧客の多様なインフラストラクチャー構成を考慮すると、ポッドが最も適切なノードにスケジュールされるようにすることです。この課題に対処するため、インテルは各ノードの機能を動的に評価する**ノード・アフィニティー・メカニズム**を実装しました。

ノードトポロジーの検出と NUMA を考慮したスケジュール

この機能は、クラスター内の各ノードのノードトポロジー検出を実行し、以下の情報を収集します。

- **numa_nodes:** Kubernetes ノード上の NUMA ノードの数。
- **cpus_per_numa_node:** NUMA ノードごとに利用可能な CPU コアの数。
- **amx_supported:** インテル® アドバンスト・マトリクス・エクステンション (インテル® AMX) がサポートされているか。通常、第 4 世代インテル® Xeon® スケーラブル・プロセッサー (開発コード名 Sapphire Rapids) 以降のプラットフォームで true となります。
- **numa_balanced:** NUMA ノードの CPU コア数とメモリー分散のバランスが取れているか。バランスの取れたトポロジーは、パフォーマンスとスケジュールの効率を向上させます。
- **max_balloons_vllm:** ノードにスケジュールできる vLLM ポッド (「バルーン」) の最大数。
- **max_balloons_reranker:** ノードにスケジュールできるリランカーポッド (「バルーン」) の最大数。
- **gaudi_available:** ノードにインテル® Gaudi® AI アクセラレーターが含まれているか。「*habana-device-plugin*」ポッドがノード上で実行されていることをチェックすることで判断されます。

この情報に基づいて、システムは各ノードに**テイントとラベル**を自動的に作成し、推論ワークロードの実行可否を判断します。vLLM や TorchServe のような LLM を提供するポッドは、インテル® AMX 対応のノードへのスケジュールを優先し、他のコンポーネントは汎用的なノードに展開される場合があります。

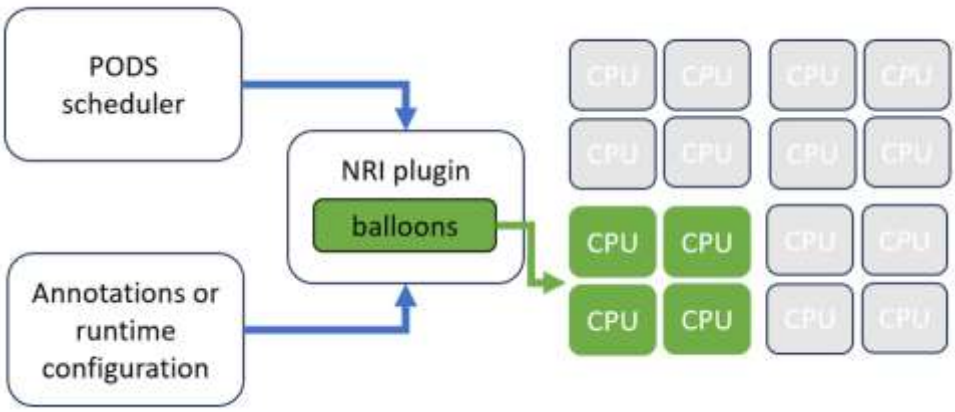
Kubernetes リソース

デフォルトでは、Kubernetes は複数のポッドが同じ物理 CPU コアを共有することを許可します。しかし、CPU 負荷の高いワークロードでは、リソースの競合を避けるため専用の CPU コアを割り当てることを推奨します。同様に、メモリー負荷の高いタスクは、パフォーマンスの低下を防ぐため、同じ NUMA ノード内に制限する必要があります。推論は大量の CPU リソースとメモリーリソースを必要とするため、Kubernetes 上でこのようなワークロードを効率良く実行することは困難です。この問題は、**ノード・リソース・インターフェイス (NRI) プラグイン** (英語) を活用することで解決されます。NRI は、CPU レベルでワークロードを分離し、基盤となるハードウェア・トポロジーに合わせて調整することで、高度な Kubernetes リソース管理を可能にします。

バルーンポリシー

この戦略の中核にあるのは、**バルーンポリシー** (英語) です。バルーン (balloons) は、ワークロードを互いに分離する論理的な CPU グループです。各バルーンは、特定のクラスのワークロード (例: 計算負荷の高い推論ポッド) に割り当てられ、重要なタスクが「うるさい隣人 (noisy neighbor)」の影響を受けないようにします。ここで「うるさい隣人 (noisy neighbor)」とは、同じリソース (CPU、メモリー、ネットワーク帯域など) を過度に消費する他のタスクを指します。

ポッドがスケジュールされると、NRI プラグインはアノテーションやランタイム構成に基づいてそれをバルーンに割り当てます。これにより、ポッドが専用の CPU コアセット上でのみ実行されることが保証され、レイテンシー、スループット、予測可能性が向上します。また、バルーンに割り当てられたリソースを他のポッドやリソースが使用できないようにすることで、厳密な分離を実現し、リソースの競合を防ぎます。



リソース分離機能

- **ノードトポロジーの検出とスケジュール:** この手順は、各クラスターノードを自動的に検査し、CPU レイアウト、NUMA 構成、CPU 世代、その他のハードウェア機能を判断します。
- **NUMA を考慮した配置:** vLLM ポッドは NUMA ノード全体に均等に分散され、クロスノード・メモリー・アクセスを回避するため、単一の NUMA ノードに制限されます。
- **兄弟 CPU 割り当て:** vLLM ポッドで使用されていない CPU は、他のワークロードに使用できます。

- **HPA による動的スケーリング:** config.yaml (英語) で *balloons.enabled* フラグが設定されている場合、水平ポッド・オートスケーラー (HPA: Horizontal Pod Autoscaler) の maxReplicas 値は、計算された maxBalloonShape に合わせて自動的に調整されます。
- **ノードごとの BalloonsPolicy 構成:** クラスター内の各ノードには、異なるポッドタイプのリソース割り当てルールを定義する独自の BalloonsPolicy オブジェクトがあります。これらのポリシーは、ノードのトポロジーと機能に合わせて自動的に作成されます。

以下は、vLLM ワークロードの構成を説明する、ノードの BalloonsPolicy の例です。

```
spec:
   agent:
     nodeResourceTopology: true
   allocatorTopologyBalancing: true
   balloonTypes:
   - name: vllm-balloon
     allocatorPriority: high
     allocatorTopologyBalancing: true
     loads:
     - llm-inference
     matchExpressions:
     - key: name
       operator: In
       values:
       - vllm
       - edp-vllm
     maxBalloons: 2
     maxCPUs: 32
     minCPUs: 32
     preferIsolCpus: false
     preferNewBalloons: true
```

主要フィールドの説明

- nodeResourceTopology: トポロジーを考慮したスケジュールを有効にします。
- allocatorTopologyBalancing: バランスの取れた NUMA 配置を保証します。
- balloonTypes: 特定のワークロード用に分離された CPU 割り当てを定義します。
- name: バルーンタイプを識別します。
- allocatorPriority: リソース割り当ての優先順位を付けます。
- loads: バルーンポリシーの別のセクションで定義されているロードクラスを指定します。
- matchExpressions: ラベル (name: vllm, edp-vllm) でポッドを一致させます。
- maxBalloons: ノードあたりの vLLM ポッドの数を制限します。
- maxCPUs / minCPUs: バルーンごとに正確に 32 個の CPU を割り当てます。
- preferIsolCpus: 分離された CPU の優先を無効にします。
- preferNewBalloons: 厳密な分離のため新しいバルーンを作成することを優先します。

オブザーバビリティーとテレメトリー

大規模なオブザーバビリティーとトラブルシューティングをサポートするため、インテル® AI for Enterprise RAG には、**Grafana ダッシュボード**と統合された堅牢なテレメトリー・スタックが含まれています。これらのダッシュボードは、システム動作、リソース使用状況、およびノード全体でのワークロード分散に関する可視性を提供し、チームがボトルネックやパフォーマンスの異常を特定するのに役立ちます。

監視に加えて、このソリューションは **Prometheus メトリック**と組み合わせて**水平ポッド・オートスケーラー (HPA: Horizontal Pod Autoscaler)** を活用し、システム負荷に動的に対応します。CPU 使用率の増加やレイテンシーなどのボトルネックが検出されると、HPA は主要なコンポーネントのレプリカ数を自動的に増やし、需要の増加に応じて持続的なパフォーマンスを確保します。

以下は、レプリカ数の経時的な変化、しきい値、測定されたメトリック値を表示する HPA ダッシュボードの例です。

このオブザーバビリティーと自動化フレームワークは、ソリューションのスケーラビリティーを検証する上で役立ちます。マルチノード・クラスターで製品のストレステストを行ったところ、複数のノードで自動的な利用率の増加が観測され、自動スケーリング、インテリジェントなスケジューリング、リソース分離、NUMA を考慮した配置の組み合わせが効率良い推論ワークロードを実現することが確認されました。

まとめ

トポロジー検出、バルーンベースの分離、および動的スケーリングの組み合わせにより、インテル® AI for Enterprise RAG ソリューションは、多様なインフラストラクチャー構成に適応可能です。ハードウェア機能と Kubernetes ネイティブのメカニズムを活用することで、スケーラブルで効率的、かつエンタープライズグレードの生成 AI アプリケーションを展開する堅牢な基盤を提供します。

法務上の注意書き

性能は、使用状況、構成、その他の要因によって異なります。詳細については、http://www.intel.com/PerformanceIndex/ (英語) を参照してください。

インテルのテクノロジーを使用するには、対応したハードウェア、ソフトウェア、またはサービスの有効化が必要となる場合があります。

絶対的なセキュリティーを提供できる製品またはコンポーネントはありません。

実際の費用と結果は異なる場合があります。

インテルは、サードパーティーのデータについて管理や監査を行っていません。ほかの情報も参考にして、正確かどうかを評価してください。