Spack を使用してインテルにより最適化された HPC バイナリーのディストリビューション

重要な HPC アプリケーションに最適な実行ファイルを構築する Spack レシピ

S. M. Iman Gohari, Ph.D. インテル コーポレーション ソフトウェア・イネーブリング & 最適化エンジニア

ハイパフォーマンス・コンピューティング（HPC）ユーザーは、パフォーマンス、互換性、精度をテストするため、異なるソフトウェアやパッケージのバージョン、依存関係、設定に対処する必要があります。この作業には時間がかかるため、ローレンス・リバモア国立研究所が主体となり、スーパーコンピューター・パッケージ・マネージャー（Spack）（英語）と呼ばれる効率的なパッケージ・マネージャーが開発されました。Spack は、その多くの利点（オープンソースで、シンプルな仕様ベースの構文を提供し、パッケージの作者 / メンテナーに同じパッケージの異なるビルドを処理する Python* インターフェイスを提供している、など）により、HPC コミュニティーの注目を集めています。Spack が 2023 年の HPCwire Editors' Choice Award の「ベスト HPC プログラミング・ツール / テクノロジー」（英語）を受賞したことは注目に値します。

この記事では、インテルのツール、ライブラリー、設定を使用してインテル® ハードウェアのパフォーマンスを最大限に引き出すことが難しい 3 つの重要な HPC アプリケーションで、最適な実行ファイルを構築する Spack レシピとガイドラインを示します。現在の作業における「最適な実行ファイル」という用語は、Spack を利用してそれらのツールを適切に使用し、インテル® HPC の顧客とパートナーに、インテル® Xeon® スケーラブル・プロセッサーで最適なパフォーマンスを達成しつつターゲットのオープンソース・アプリケーションを簡単に構築する一般的な方法を提供することを指します。これらのレシピは、Amazon Web Services*（AWS*）および Google Cloud

Platform*（GCP*）上の第 4 世代インテル® Xeon® スケーラブル・プロセッサーでテストされ、インテルにより最適化され、手動でビルドされたバイナリー（以下、「リファレンス・ビルド」と呼びます）と同等のパフォーマンスが得られています。

ターゲット・アプリケーション

HPC アプリケーションは、科学、エンジニアリング、ビジネスにおける複雑な問題を解決します。Spack を利用可能なパッケージのリスト（英語）は増え続けています。産業、研究、学術分野で広く使用されているため、さまざまな HPC 業界にわたる、次のアプリケーションが現在の作業に選択されています。

- Weather Research and Forecasting Model（WRF）（英語）は、大気のモデリングと運用予測のために設計された数値気象予測モデルです。WRF のビルドプロセスは、一連の依存関係が含まれているため、「かなり」複雑なものと考えられています。そのため、ビルドを簡素化する Spack の能力を評価するのに役立ちます。
- Large-scale Atomic/Molecular Massively Parallel Simulator（LAMMPS）（英語）は、材料のモデリングに重点を置いた古典的な分子動力学シミュレーターですが、異なるスケールの汎用粒子シミュレーターとして機能します。LAMMPS のビルドプロセスは簡単ですが、そのパフォーマンスはコンパイラー、命令セット、パフォーマンス・ライブラリーなどの選択に左右されます。そのため、ビルドプロセス中に発生する潜在的なパフォーマンスの低下を追跡するのに役立ちます。
- Open Source Field Operation and Manipulation（OpenFOAM）（英語）は、数値のモデリング、および主に数値流体力学向けの連続体力学の前処理と後処理のためのオープンソースのツールボックス・スイートです。コンパイラー間の依存関係があるため、コンパイルプロセスに比較的時間がかかります。そのため、コンパイラー間の依存関係や長いコンパイルを処理する Spack の能力を評価するのに役立ちます。

システム構成

クラウド・サービス・プロバイダー（CSP）（AWS* や GCP* など）は、オンプレミスのシステムで発生する初期コストや導入の困難さを回避しながら、アプリケーションとデータを処理するオンデマンドで、オープンアクセスの、スケーラブルなコンピューティング・リソースをユーザーに提供します。この記事では、公開されている CSP のインスタンスを使用して Spack レシピのパフォーマンスを評価し、読者がレシピを再現できるようにしています。**表 1** と**表 2** はそれぞれ、AWS* と GCP* のインスタンスの概要です。提供された Spack ガイドラインが包括的で有用であり、異なるインテル® アーキテクチャーでテストされていることを保証するため、2 つの CSP で提供されている 4 つの異なる世代のインテル® Xeon® プロセッサーを含むインスタンスでテストを行いました。

システム構成	AWS* c5n.18xl	AWS* c6i.32xl
テスト実施者	Intel Corporation	Intel Corporation
CSP/ リージョン	AWS* us-east-2	AWS* us-east-2
インスタンス・タイプ	c5n.18xl	c6i.32xl
vCPU 数	72	128
ソケット	2	2
コアごとのスレッド数	2	2
NUMA ノード数	2	2
反復数と結果の選択	上位 3 つの反復の中央値	上位 3 つの反復の中央値
アーキテクチャー(開発コード名)	Skylake	Ice Lake
メモリー容量	192GB 2666MT/s	256GB 3200MT/s
SKU	インテル® Xeon® Platinum 8124M	インテル® Xeon® Platinum 8375C
ストレージタイプ	Amazon* FSx Luster	Amazon* FSx Luster
OS	CentOS* 7 (Core)	CentOS* 7 (Core)
カーネル	kernel-3.10.0-1160.88.1.el7.x86_64	kernel-3.10.0-1160.88.1.el7.x86_64
参考資料	Amazon* EC2 C5 インスタンス	Amazon* EC2 C6i インスタンス

表 1. AWS* インスタンスのシステム構成の詳細

システム構成	GCP* c2-stand-60	GCP* c3-highcpu-176
テスト実施者	Intel Corporation	Intel Corporation
CSP/ リージョン	GCP* us-central1	GCP* us-central1
インスタンス・タイプ	c2-stand-60	c3-highcpu-176
vCPU 数	30	176
ソケット	2	2
コアごとのスレッド数	2	2
NUMA ノード数	2	4
反復数と結果の選択	上位 3 つの反復の中央値	上位 3 つの反復の中央値
アーキテクチャー(開発コード名)	Cascade Lake	Sapphire Rapids
メモリー容量	240GB	352GB
SKU	インテル® Xeon® CPU @ 3.10GHz	インテル® Xeon® Platinum 8481C @ 2.70GHz
ストレージタイプ	標準永続ディスク	標準永続ディスク
OS	CentOS* 7 (Core)	CentOS* 7 (Core)
カーネル	3.10.0-1160.81.1.el7.x86_64	3.10.0-1160.81.1.el7.x86_64
参考資料	C2 マシンシリーズ	C3 マシンシリーズ

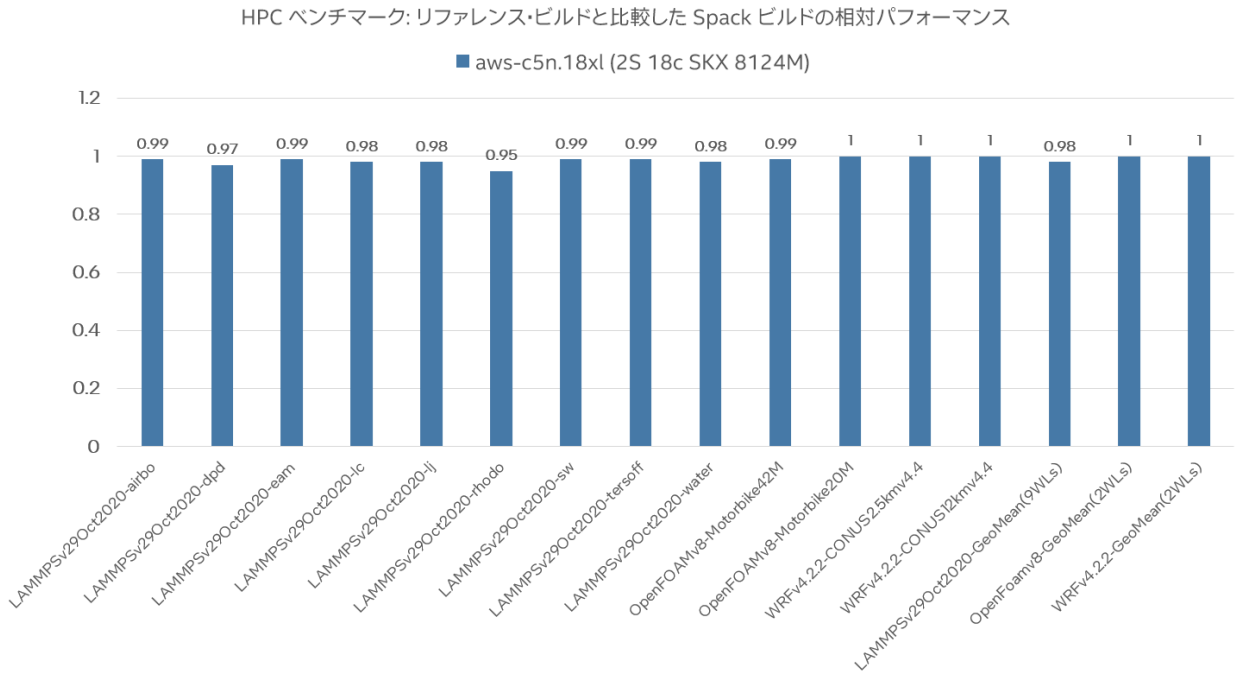
表 2. GCP* インスタンスのシステム構成の詳細

パフォーマンス解析

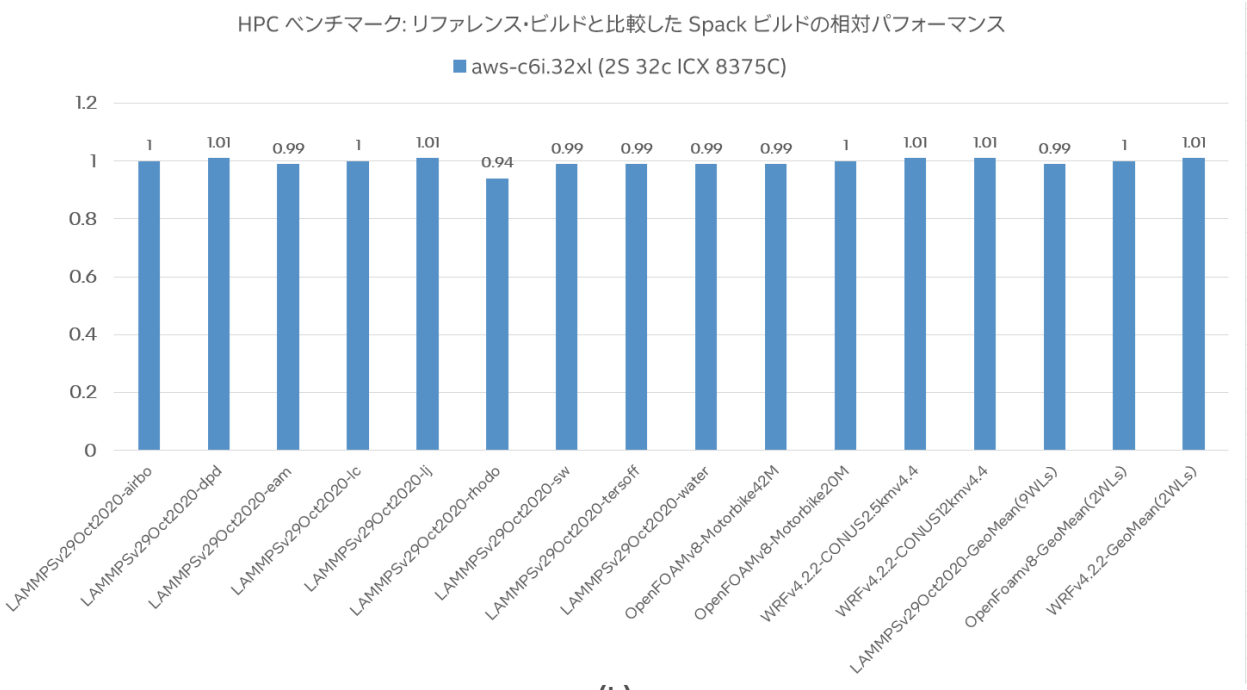
前述の 3 つのアプリケーションの 13 のワークロードのパフォーマンスを、次の点を考慮して評価しました。

- Spack バイナリーを対応するリファレンス・ビルドと比較する際、同様の（または同じ）コンパイラーやアプリケーションのバージョンを利用して、ソフトウェア・スタックの変更を最小限に抑える。
- 同じベンチマーク・ツール、環境変数 / 設定、ワークロード・タイプ / バージョン、入力データセット、その他を使用して、各ビルドのパフォーマンスを測定する。

図 1 は、AWS* インスタンス上の対応するリファレンス・ビルドと比較した Spack ビルドのパフォーマンスを示しています。実行ごとの変動に伴う 10% の許容誤差を考慮すると、リファレンス・ビルドの予想パフォーマンスが、調査したすべてのワークロード間で Spack ビルドを使用して（ほぼ）再現されています。同じテストと実行プロセスを GCP* のインスタンスで実行したときの、対応するパフォーマンス結果を**図 2** に示します。繰り返しますが、リファレンス・ビルドと Spack バイナリーのパフォーマンスはほぼ同等です。

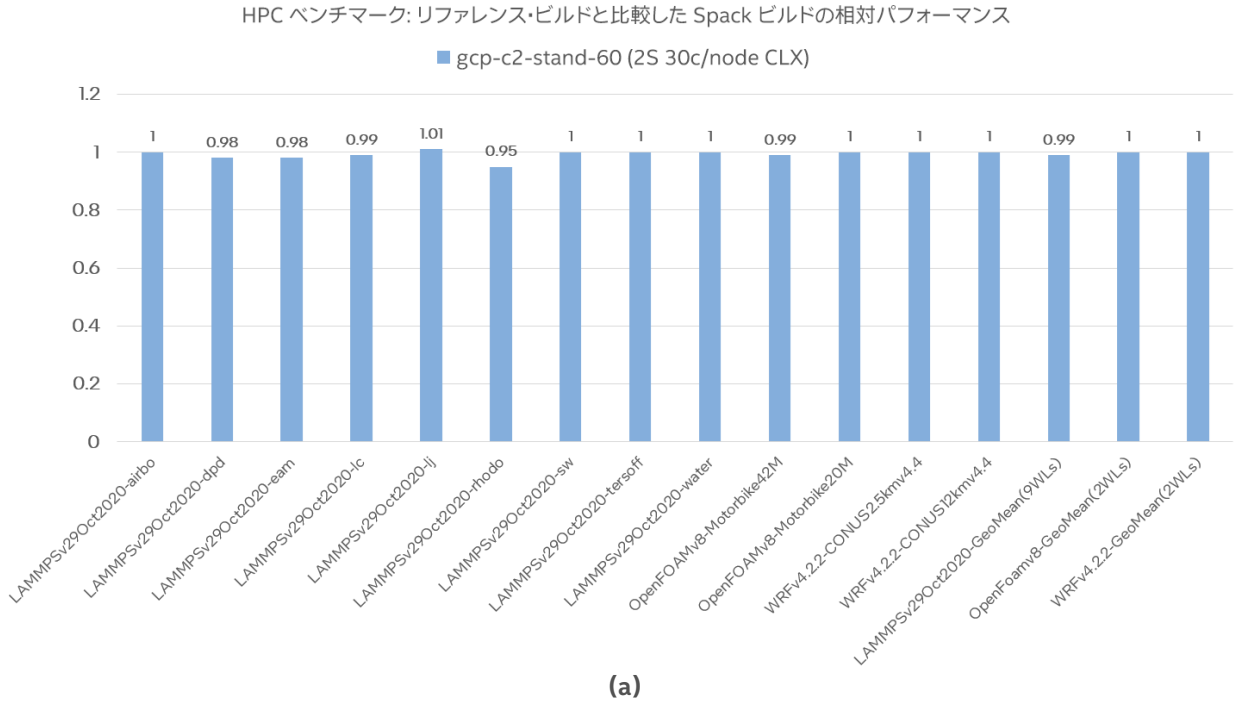


(a)

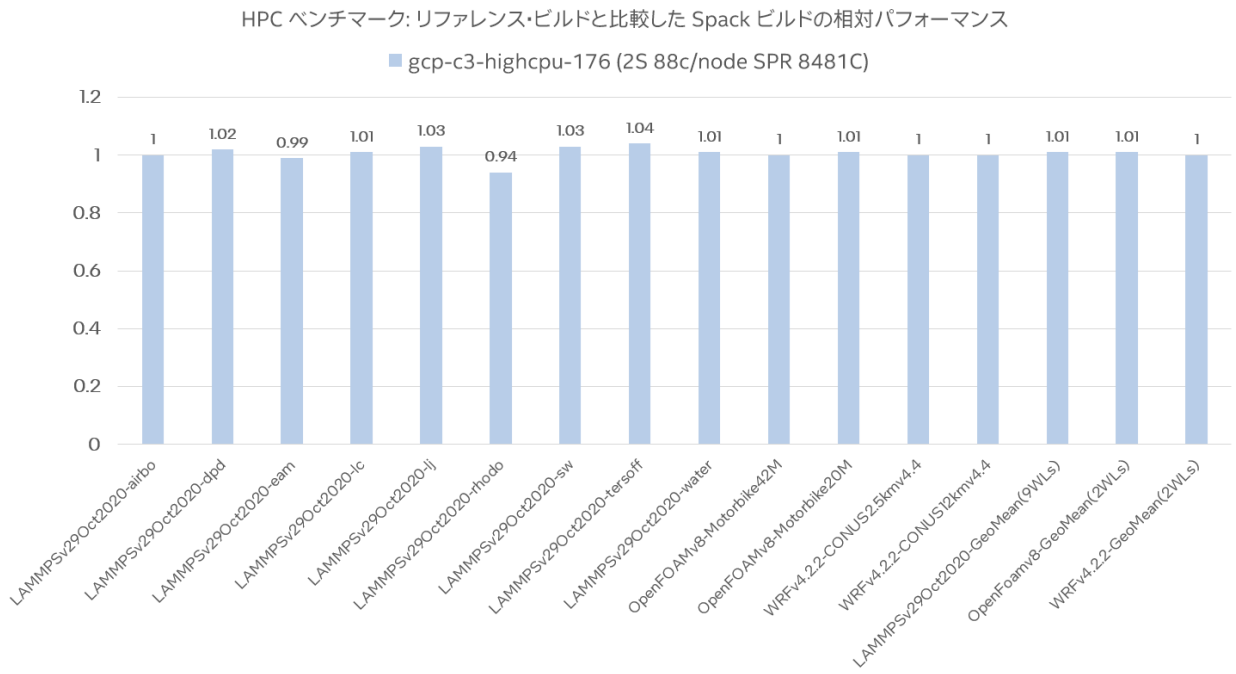


(b)

図 1. AWS* でテストした HPC ベンチマークの Spack ビルドとリファレンス・ビルドの相対パフォーマンス：(a) c5n.18xlarge、(b) c6i.32xlarge



(a)



(b)

図 2. GCP* でテストした HPC ベンチマークの Spack ビルドとリファレンス・ビルドの相対パフォーマンス : (a) c2-stand-60、(b) c3-highcpu-176

調査対象の HPC アプリケーションには、リファレンス・ビルドと比較するための客観的に定義された「ベースライン」構成が存在しないことに注意してください (WRF のコンパイルには 70 を超えるオプションが用意されているためです)。主観的な結論を避けるため、「インテルにより最適化されたバイナリーとベースライン」の比較は行っていません。

Spack レシピ

HPC アプリケーションのエンドツーエンドのインストールは、システム・アーキテクチャー、依存関係の可用性、アプリケーションのコンパイル時間などによって異なります。Spack は、そのようなインストール (および必要なすべての依存関係) を合理化することにより、「使いやすさ」を提供します。手動プロセスと比較してエンドツーエンドの時間のオーバーヘッドはありません。**表 3 ~ 5** は、v0.20.0 リリース (英語) を使用して LAMMPS、OpenFOAM、WRF のパフォーマンス解析セクションの結果を取得するために使用した Spack レシピの概要です。これらの重要な HPC アプリケーション向けの最適なバイナリーを構築するには、Spack レシピ (または同様のもの) を使用することを推奨します。Spack レシピを使用する前提条件として、基本的な Spack インストール (英語) とインテル® oneAPI パッケージのインストール (英語) が必要なことに注意してください。

アプリケーション	LAMPS
ビルドコマンド	$spack install lammps@20201029 +asphere +class2 +kspace +manybody +misc +molecule +mpiio +opt +replica +rigid +user-omp +user-intel %intel@2021.6.0 ^intel-oneapi-mkl ^intel-oneapi-mpi target=skylake_avx512
バージョンとワークロード	V20201029。airebo、dpd、eam、lc、lj、rhodo、sw、tersoff、water
ノート	Spack のソースコードは変更していません。

表 3. LAMMPS の最適なビルド向けの Spack レシピ

アプリケーション	OPENFOAM
ビルドコマンド	$spack install openfoam-org@8%intel@2021.6.0 ^intel-oneapi-mpi target=skylake_avx512 ^scotch%gcc
バージョンとワークロード	V8。 MotorBike 20M（250 反復）、MotorBike 42M（250 反復）
ノート	Spack のソースコードは変更していません。

表 4. OpenFOAM の最適なビルド向けの Spack レシピ

アプリケーション	WRF
ビルドコマンド	$spack install wrf@4.2.2%intel@2021.6.0 build_type=dm+sm ^intel-oneapi-mpi ^perl%intel ^libxml2%intel ^tar%oneapi@2022.1.0 ^jasper@2.0.32 ^m4%gcc@8.5.0
バージョンとワークロード	V4.2.2。CONUS-12km（v4.4）、CONUS-2.5km（v4.4）
ノート	WRF のタイミングを改善する推奨オプション（英語）を Spack ソースコード（英語）に反映できます。推奨オプションは精度に影響を与える可能性があることに注意してください（詳細（英語））。

表 5. WRF の最適なビルド向けの Spack レシピ

まとめ

HPC アプリケーションは、さまざまな複雑な問題を解決するのに役立ちますが、アプリケーションの構築とテストのプロセスは、単調でコストがかかり、時間もかかります。Spack は、HPC パッケージのユーザーに、自動構築と依存関係を追跡するシンプルな、オープンソースの、仕様ベースのインターフェイスを提供します。この記事では、Spack を使用して、最適なパフォーマンスを達成しつつ、重要な HPC アプリケーションを簡単に効率良く構築できることを説明しました。パフォーマンス解析は、AWS* と GCP* で動作している 4 つの異なる世代のインテル® Xeon® プロセッサー上で行いました。この記事で説明した Spack レシピ（または同様のもの）を使用して、紹介した HPC アプリケーション向けの最適なバイナリーを構築することを推奨します。

その他の Spack の資料（英語）

- Spack : HPC ソフトウェアのための柔軟なパッケージ・マネージャー
- GitHub* の Spack リポジトリー
- チュートリアル : Spack 101
- Spack のドキュメント
- Spack のインテル® oneAPI パッケージ