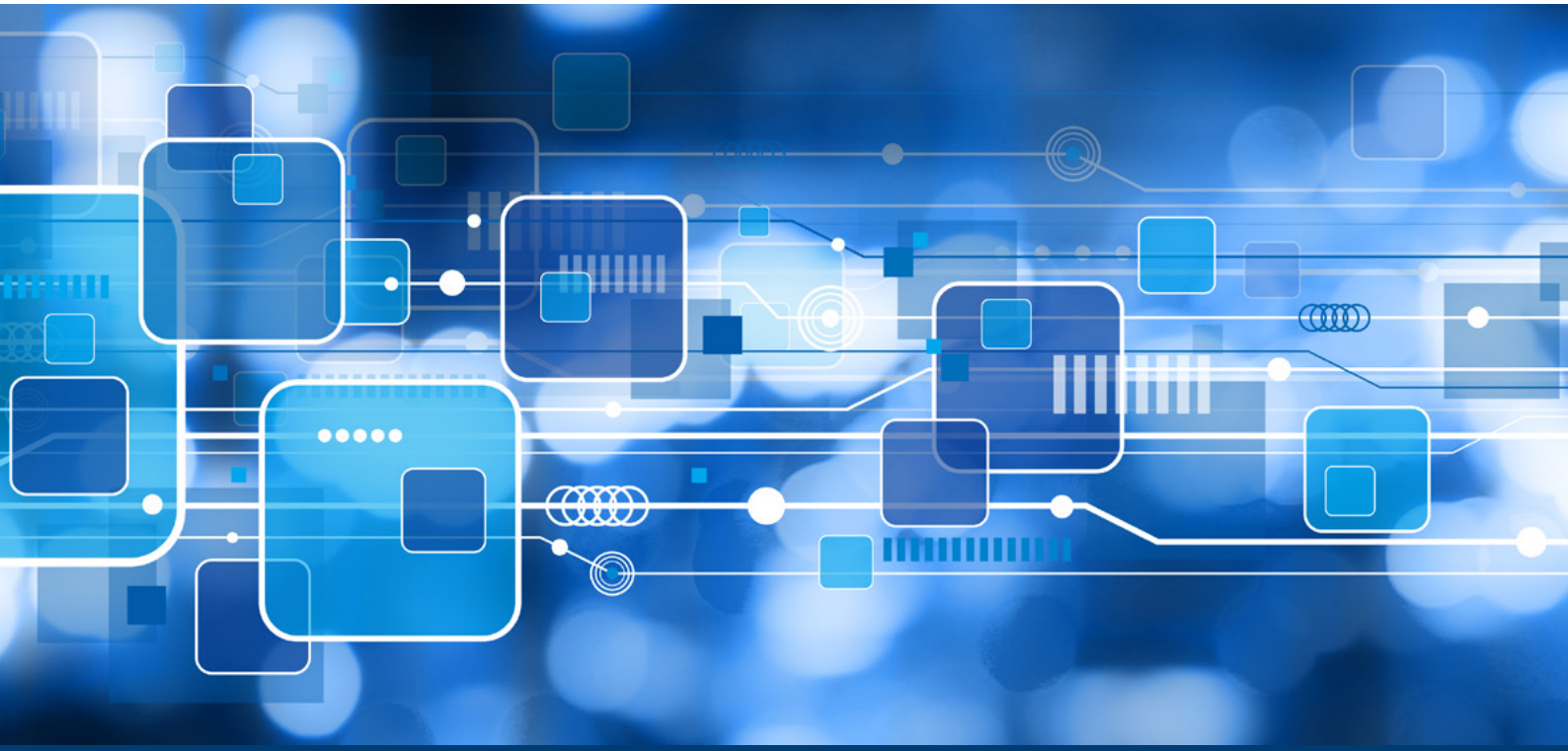


インテルと ANACONDA* によるオープン・データ・サイエンス向け PYTHON* の強化

ビッグデータの課題への取り組みを保証するテクノロジー

Travis Oliphant Continuum Analytics 設立者兼 CEO

現在、最も注目されている計算分野は、ハイパフォーマンス・コンピューティング (HPC)、ビッグデータ、データサイエンスでしょう。しかし、最近まで、これらの分野には関連性はほとんどありませんでした。ハードウェアの進歩は HPC における大躍進の原動力となりましたが、オープン・データ・サイエンス (**ODS**) では、Python* および R* コミュニティーからのフィードバックに依存しているのが現状です。

最適化を行わないと、Python* のような高水準言語は、大きなデータセットの解析に必要なパフォーマンスが得られません。幸いなことに、**Continuum Analytics** (**Anaconda*** の設立者および作成者) は、最先端の計算速度と開発の容易さを結び付ける方法を見つけました。

Anaconda* は、**インテル® マス・カーネル・ライブラリー** (インテル® MKL) がリンクされたハイパフォーマンスな Python* ディストリビューションを含む、優れた **ODS** プラットフォームです。Python* を始めたばかりの開発者でも利用可能な、ビッグデータ・プロジェクトへの取り組みを支援する強力なパッケージとテクノロジーを提供します。Anaconda* は、高度な解析、数値計算、JIT コンパイル、プロファイリング、並列処理、対話型の視覚化、コラボレーション、その他のニーズをサポートします。

最近の HPC の進歩と Python* ベースのデータサイエンスを組み合わせることにより、Anaconda* コミュニティーは、業界の将来を特徴付けるビッグデータの課題への取り組みを保証する高性能の (低コストで使いやすい) 解析テクノロジーの開発に取り組んでいます。

ODS と Python* でデータサイエンスの革新を加速

科学計算およびデータサイエンスの世界では、2 つのオープンソース言語、Python* および R* が利用されています。これらの言語は、ドメイン・エキスパート (金融アナリスト、データ・サイエンティスト、データエンジニア、ビジネスアナリストなど) に、熟練したプログラマーになることなく、解析ニーズに集中できる環境を提供します。

Python* は特に親しみやすい言語です。世界中のコンピューター・サイエンスの教室で教えられ、コミュニティーで活発に議論されています。Python* は、シンプルで読みやすいコードを記述すること (「Pythonic」であること) を特に重視しています。「**Zen of Python**」でも述べられているように、「シンプルは複雑より良い」のです。

Python* はデータサイエンスに理想的な言語です。Anaconda* は、Python*、R*、Hadoop* および成長を続けるオープンソース・テクノロジーのパフォーマンスを最大化する強固な ODS プラットフォームを提供します。Anaconda* は、さまざまなアーキテクチャー (Raspberry Pi* からハイパフォーマンス・クラスターまで) においてスケールアップとスケールアウトを実現し、企業の変化するパフォーマンス・ニーズを満たす柔軟性を備えています。

Python* には、「バッテリー同梱 (batteries included)」の哲学もあります。一般的なプログラミング・タスクを行うためにサイズの大きな標準ライブラリー・パッケージが用意されており、ポリシー・モデリングから化学まで、さまざまな分野に特化した多くのサードパーティー・ライブラリーと統合されます。Anaconda* プラットフォームは、これらの標準パッケージを含んでいるだけでなく、パッケージを簡単に追加できます。Python* を使用するデータ・サイエンス・チームは、(パフォーマンスや使いやすさを犠牲にすることなく) 思いつくほぼすべてのものをモデル化することができます。プラットフォームは単純なインストールでそのまま利用できる機能を提供するため、これらのパッケージはデプロイ環境で簡単にセットアップすることができます。

単純性、寛容さ、パフォーマンスを備えた Python* と Anaconda* の組み合わせは、ドメイン・エキスパートやデータ・サイエンティストに (革新をもたらす) さまざまなモデルをテストおよび操作する環境を提供します。インテル コーポレーションのチーフ・エバンジェリスト James Reinders は、「プログラミングを主要な作業であると考えないデータ・サイエンティスト、化学者、物理学者にとって、プログラミングは弊害となります。Python* のような環境を利用することは、必要なパフォーマンスを実現する合理的な投資であり、科学者はプログラミングではなく科学により集中することができるのです。」と述べています。

ありがたいことに、ドメイン・エキスパートには、ODS で今日のハードウェアの計算能力を利用できるように支援する多くの Python* ベースのオプションがあります。下記に説明するテクノロジーはすべて、Anaconda* Enterprise サブスクリプションに含まれており、開発環境に効率良く統合されます。また、これらはすべて、科学者、経済学者、財界人を含む、プログラマーでない人でも利用できます。

高速でシンプルなコードを実現する Python* のコンパイル

ドメイン・エキスパートが便利さを実感する Anaconda* の機能の 1 つは、ほかのすべての Python* 解析のベースとなる基本的な Python* 解析パッケージ、NumPy* に統合される Python* 用の Numba* 実行時 (JIT) コンパイラーです。Numba* は、ユーザーが Python* コードのセクションを Python* デコレーター (関数の処理を修飾する機能) でマークできるようにします。マークしたセクションはマシンコードにコンパイルされ高速に実行されます。

Python* の解釈に依存する代わりにコンパイラーが生成したネイティブ・マシンコードを使用することで、Python* の実行速度は 2 倍から 2,000 倍になります。パフォーマンスの向上に加えて、Numba* ではコンテキストの切り替えが必要ないため、プログラマーの精神的な負担も軽減されます。つまり、Numba* ユーザーは、パフォーマンス・クリティカルなセクションを C/C++ や Fortran のような従来のコンパイル型言語に変更することなく、Python* エコシステム内ですべての操作を行うことができるのです。Anaconda* と Numba* で NumPy* コードを最適化および単純化した、最新のオープンソース米国税金ポリシー・モデリング・プロジェクト **TaxBrain*** (**taxcalc*** パッケージを使用する Web アプリケーション) は、Anaconda* の潜在能力を示す優れた例の 1 つです。

また、Anaconda* は、データ・サイエンス・チームが並列化により計算パフォーマンスを向上できるように支援します。単一行のデコレーターを挿入することにより、ユーザーは指定したスレッド数で実行するように関数に指示することができます。複数のスレッドに分割できるタスクの場合、Numba* は最近のプロセッサー (GPU を含む) の複数のコアを利用する非常に簡単な方法をプログラマーに提供します。例えば、Numba* で 1 行のコードを追加して 8 つのコアで同時に実行するように関数を修飾すると、速度は 8 倍になります。

Python* がアルゴリズムおよびデータ・サイエンス・アプリケーションの迅速な開発に理想的とされるのは、これらの速度の利点があるためです。「Python* には、迅速な対話型開発用のさまざまなツールが用意されています。その**並列計算**機能は比類なきものであり、Anaconda* およびインテル® MKL と組み合わせることで、Python* は開発者がマシンラーニングやディープラーニングのアルゴリズムの新しい境地に到達することを支援できるでしょう。」(James Reinders)

Anaconda* は、ほかの HPC ツールとの相互運用性を向上するため、活発に開発を行っています。現在進行中のプロジェクトの 1 つは、**インテル® スレッディング・ビルディング・ブロック** (インテル® TBB) ライブラリーのインターフェイスを作成することです。このような相互運用性は、ODS の活動の特徴です。

インテル ® C++ コンパイラー (インテルベースのハードウェアのパフォーマンスをさらに向上) および将来も利用できる Python* スクリプト (新しいインテルの技術革新を利用してスケーリング) を使用する Anaconda* のパフォーマンス分散プロジェクトも進行中です。

Anaconda* とインテル® MKL の組み合わせによる Python* の重要な解析パッケージの最適化

Anaconda* は、最速の計算およびスループットのために最適化された、プリコンパイル済みのライブラリーを提供するため、インテル® MKL により Python* の数値計算パフォーマンスを強化し、NumPy*、SciPy*、scikit-learn、Numexpr* を含む、最も一般的な Python* 数値解析ライブラリーの土台となる線形代数および高速フーリエ変換 (FFT) の演算性能を向上させました。インテル® MKL は、数値解析を 10 倍から 100 倍高速化します。

NumPy* には、少数の要素にのみ定義された基本的なカーネル関数を非常に大きな配列に対して自動的にマップする重要な機能があります。この基本的なカーネルは、典型的な NumPy* 関数呼び出しで何度も実行されます。基本的なカーネルは、通常、コンパイルされたコード (C、C++、Fortran) とともにビルドされるため、NumPy* を本来の速度で実行できます。Anaconda* と Python* コンパイラーを利用することにより、これらの基本的なカーネルを Python* で記述し、単純な Python* 構文を使用して基本的なレベルで NumPy* を簡単に拡張することができます。これらのカーネルには、さらに多くの機能を提供するインテル ® MKL 機能への呼び出しを含めることもできます。

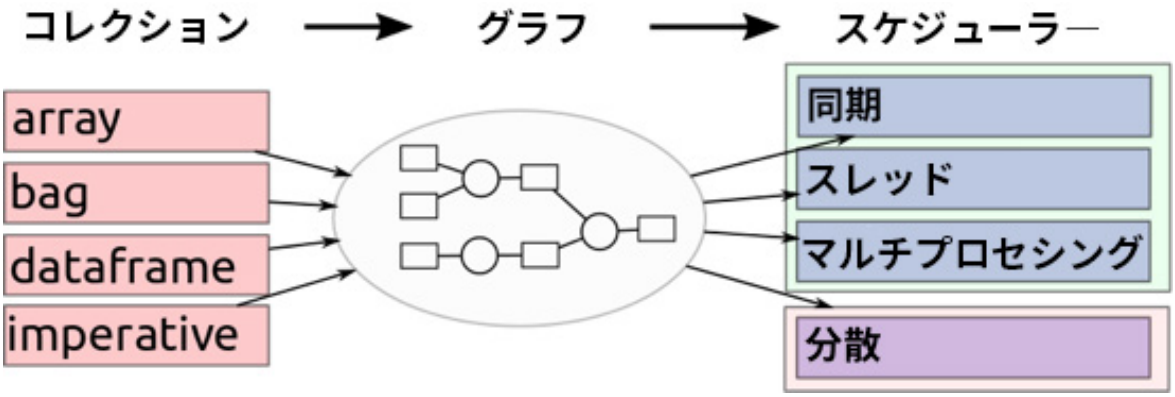
Anaconda* はインテル® MKL により拡張された多くの基本的な NumPy* 関数を提供するため、ユーザーは、三角法、平方根、代数関数呼び出しを含む、一般的に使用される関数をそのまま利用して、優れたパフォーマンスを得ることができます。「インテル® MKL のおかげで、今日の NumPy* および SciPy* ユーザーは、1 行のコードも変更することなく、アプリケーションのパフォーマンスを大幅に向上することができます。」(James Reinders)

ODS の強力で柔軟な分散コンピューティング

パフォーマンスをさらに向上するため、Anaconda* を使用するデータ・サイエンス・チームは、高速な数値解析のために低オーバーヘッドと低レイテンシーおよび最小限のシリアル化で動作する、強固かつハイパフォーマンスな並列計算フレームワークを活用できます **(図 1)**。HPC および Hadoop* クラスターで解析ワークロードの柔軟な分散処理が可能です。

1 並列化の種類

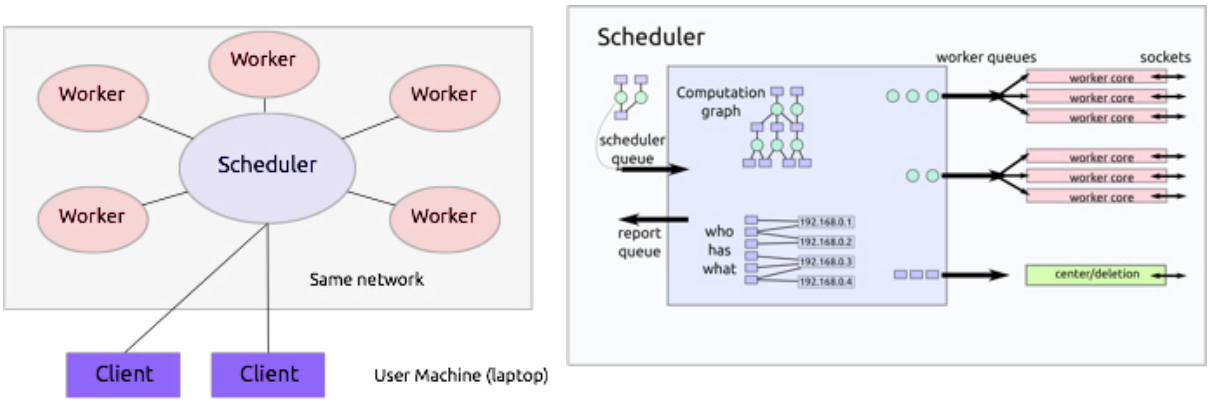
Anaconda* は、計算用によく知られた配列形式またはデータフレーム・コードを記述するための高水準インターフェイスをデータ・サイエンス・チームに提供します **(図 2)**。新規または既存の Python* スクリプトは、タスクの有向非巡回グラフ (DAG) に変換されます。これらの DAG は、大きなワークロードを多くのより小さな処理タスクに分割します。DAG は、ビルトインまたは独自のタスク・スケジューラーにより実行できます。Anaconda* には、単一マシンまたは HPC や Hadoop* クラスターのノードにわたるマルチスレッド、マルチプロセシング、分散処理に対応したスケジューラーが用意されています。クラスターは、クラウドベースまたはベアメタルのクラスターになります。DAG は、マルチコア CPU でよく知られた配列またはデータフレーム操作を使用して、数十 GB あるいは数百 GB のデータを非常に素早く処理することができます。



2 Anaconda* の並列計算フレームワーク

パフォーマンスの利点は、最近のハードウェアに非常に適したタスクの配列計算と自動チャンクによるものです。例えば、Anaconda* は、1,000 万行の配列を 1,000 万行のセグメントに分割するため、DAG は大きな配列チャンクの各ワークの関数に対して多くの呼び出しを行います。このアプローチは、金融計算では非常に一般的です。Anaconda* は、この手法を広範に適用します。DAG は Python* ディクショナリーとして表現されるため、データ・サイエンス・チームは DAG を柔軟に変更できます。必要に応じて、チャンクを変更したり、呼び出しを挿入します。次に、グラフは、マルチスレッド、マルチプロセシング、マルチノードの分散処理のために、3 つのビルトイン・スケジューラーのいずれかに送られます。既存のスケジューラーが解析に適していない場合、Anaconda* の開発者はゼロからまたはビルトイン・リファレンスを変更して、独自のスケジューラーを作成することができます。

図 3 は、分散型スケジューラーの例です。



3 分散型スケジューラーの例

単一マシンでは、Anaconda* はストレージまたは共有メモリーからデータを処理して、最近の CPU のコアをすべて使用します。クラスターでは、Anaconda* はデータ・サイエンス・チームが多くのノードに新しいアルゴリズムをプロトタイプして配備できるようにします。テキスト処理、統計、マシンラーニング、画像処理では、PySpark* の 10 倍から 100 倍高速化されます。

図 4 は、マシンラーニング・パイプラインの例です。

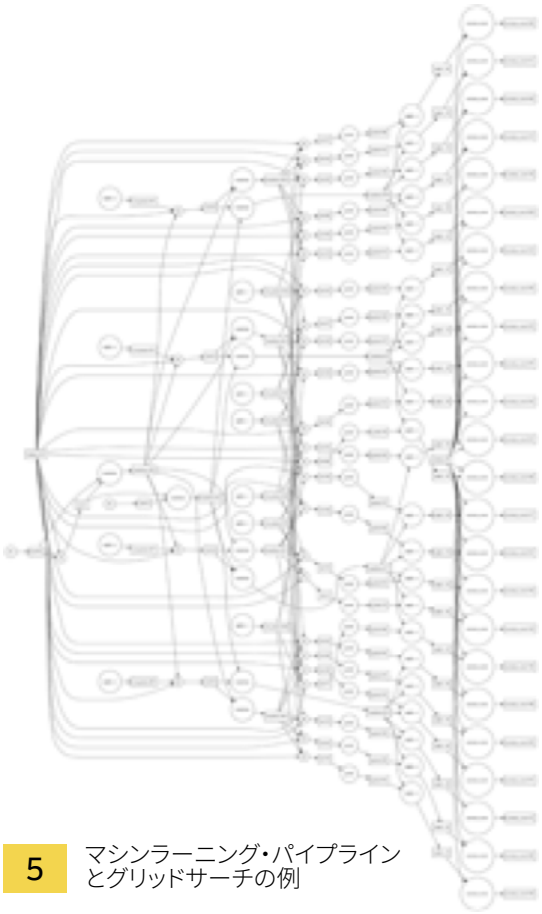
4 マシンラーニング・パイプラインの例

並列計算フレームワークは、データサイエンスに特に適しています。NumPy* と同じ API および一般的な **pandas*** ライブラリーを使用します。同じ API 呼び出しで、コードを並列に実行できます。例えば、メニーコアのインテル® Xeon Phi™ コプロセッサーを 2 つの並列メソッド呼び出しで使用している場合、Anaconda* は解析速度を何倍にもすることができます。幸い、利用可能なすべての高水準手法において、Anaconda* は簡単に使用でき、利点を得るために並列処理の知識は必要ありません。

図 5 は、マシンラーニング・パイプラインとグリッドサーチの例です。

データ・サイエンス・チームは、配列コンピューティング形式で 1 回コードを書くことにより、Anaconda* でハードウェアの能力を簡単に引き出すことができます。Anaconda* は、このコードを並列化して、コアまたはノードの処理速度を自動的に最大化することができます。「PyData」スタックで構築された Anaconda* の並列計算フレームワークは、気候科学 (xarray)、画像処理 (scikit-image)、ゲノミクス (scikit-allel)、マシンラーニング (scikit-learn) を含む、さまざまな分野向けに設計されたパッケージの速度を向上します。

データとワークロードの変化によって生じるボトルネックや競合ポイントの特定には、Anaconda* のプロファイラーと診断が役立ちます。見つかったボトルネックは、DAG を変更して修正することができます。この方法では、最初の DAG をベースにして、常に変化するデータやクラスターに合わせて、迅速かつ柔軟に対応することができます。並列計算フレームワークは、弾力性があり伸縮自在です。含まれているリソース・スケジューラー (マルチスレッド、マルチプロセシング、分散処理) は、Torque*、Slurm*、YARN* を含むサードパーティーのスケジューラーと統合できます。この柔軟性とパフォーマンスにより、Anaconda* は最良の価格 / 性能比でより短期間に価値を産み出せるようにインフラ投資を最大化します。



5 マシンラーニング・パイプラインとグリッドサーチの例

プロプライエタリーなソリューションに対する利点により、オープンソース・ソフトウェアの利用は増加しています。

インテル® VTune™ Amplifier XE と Dask.diagnostics* を使用した Python* のプロファイリング

データ・サイエンス・チームがパフォーマンス問題に取り組む最良の方法の 1 つは、プロファイリングを利用することです。インテルと Continuum Analytics は、Python* コードのプロファイリングを支援する機能を提供しています。

関数のプロファイリングについては、関数のタイミングデータを収集する cProfile が Python* の標準ライブラリーにすでに含まれています。SnakeViz (cProfile のモジュールの出力を表示するブラウザーベースのグラフィカル・ビューアー) は、プログラマーがボトルネックを特定できるようにパフォーマンス情報を表示します。ユーザーは、heapy および line_profile パッケージを使用して、Python* のメモリーおよび行プロファイリングを実行することもできます。

これらはパフォーマンス・ボトルネックを特定する便利なツールですが、配列やデータフレームを含む大規模な計算のプロファイリング中に、プログラマーがパフォーマンス問題を解決する手助けとはなりません。多くのプロファイリング・ツールは関数の実行にかかる時間を示しますが、その関数に渡された引数の詳細情報を提供することはまれです。例えば、引数がスカラー、ディクショナリー、配列であることは示されますが、配列のサイズ、型、形状は示されません。**インテル® VTune™ Amplifier XE** のような高度なプロファイリング・ツールは、ロックで待機しているスレッドを含め、Python* およびネイティブコードの行レベルで特定の関数がどのように呼び出されているか詳細に示します。多くのビッグデータ・アプリケーションでは、パフォーマンス hotspot やボトルネックを診断する際にこれらの情報が重要になります。

Anaconda* はインテル® MKL により拡張された NumPy* 関数を提供するため、ユーザーは優れたパフォーマンスを得ることができます。

この種のプロファイリングは、クラスターの並列計算ではさらに困難になります。パフォーマンス・ボトルネックの影響を評価できるように適切な診断および可視性を得ることは、データ・サイエンス・チーム全体にとって重要です (特にクラスターでスケーリングする場合)。

幸い、クラスターの並列コードのプロファイリングは Anaconda* でも行うことができます。Anaconda* には、この困難なタスクを支援するためのさまざまな機能が用意されています。スケジューラーのコールバックで分散スケジューラーの実行を調査できます。進捗バーは長い計算の完了状況を示します。分散スケジューラー用のプロファイラーは、スケジューリング、リソース、キャッシュ (開始時間と終了時間など)、メモリー、CPU 使用率、サイズメトリックおよびその他の値についての情報を提供します。

価格 / 性能比と HPC データサイエンス

データ・サイエンス・チームが (プロプライエタリーまたは ODS ソリューションの) どちらのソフトウェア・エコシステムを選択した場合でも、コストが問題になります。かつて、データストレージはコストを左右する大きな要因でした。しかし、現在では、ストレージの 1 テラバイトあたりの価格は大幅に低下しています。

このコストの低下により、ビッグデータを扱えるようになりましたが、データ処理コストが無関係になったわけではありません。今日多くの場合、価格 / 性能比でこれらのコストが評価されます。多くのチームは、この比率を低く抑え、安価に格納した大量のデータから有用な結果を得ようとしています。

データ・サイエンス・チームがシステムの価格 / 性能比を低く抑える 1 つの方法は、ODS を採用することです。このアプローチは、ハードウェアで多くの進歩があった場合にも引き続き利用できます。ハードウェアの処理能力が増加するとともに、パフォーマンス特性も向上します。企業がより新しいハードウェアで既存のソフトウェアを実行できる場合、全体的なコストが削減され、価格 / 性能比は低くなります。新しいハードウェアがリリースされたときに新しいソフトウェアを購入すると、新しいソフトウェアのライセンス料、トレーニング・コスト、開発者がシステムを学び直したりコードを書き直したりするために必要な時間が発生するため、この比率は高くなります。

ODS は、価格 / 性能の面で利点があります。これまで紹介したテクノロジーの多くはオープンソースです。無料で利用でき、相互運用可能で、オープンソース・コミュニティーにより絶えず更新されています。これらのオープンソースのツールはプロプライエタリーでないため、ベンダーのリリース・スケジュールやライセンスに制限されません。

これらのプロプライエタリーなソフトウェアのコストは、解析製品のライセンスの購入や更新によるものです。また、コンサルタントやアドオンパッケージにもコストがかかります。プロプライエタリーなソフトウェアにかかるコストは増えることはあってもなくなることはありません。

一方、オープンソース・ソフトウェアを利用すると、これらのコストの多くは大幅に減るか不要になります。これまでに紹介したものも含め、多くの ODS パッケージは無料で簡単にインストールできます。さらに、プロプライエタリーなソフトウェアのライセンスおよび更新に相当するコストは発生しません。ODS コミュニティーが無料のサポートを提供しているため、法外なコンサルタント費用が含まれることはまれです。

誤解のないように言っておきますが、ODS はすべてのコストを消し去る魔法の杖ではありません。ODS アプローチを採用した場合でも、オープンソース・パッケージを使用した SaaS 製品には開発者のコストがかかります。もちろん、ハードウェア・コストも、解析の総コストに含まれます。

多くの場合、解析に対する純粋なオープンソースのアプローチには予期しない問題も発生します。ODS ベンダーによるサポートがない場合、ソフトウェアをセットアップおよび保守するエキスパートが必要になるため、ソフトウェアの実装コストは跳ね上がります。満足するソリューションを提供できるエキスパートを見つけることは困難です。

変化の速度

これは業界全体で進行中の大きな変化です。TaxBrain* のような公共部門のプロジェクトは、ODS がビジネス、政府、ソーシャルグッドの変化に大きな役割を果たしていることを示しています。金融サービスでは、クオンツ、トレーダー、デベロッパー、アナリストで構成されるさまざまなチームで、ODS ソリューションを利用しています。

ビジネス界では、Anaconda* による対話型視覚化の進歩によって、多くの企業がビッグデータ・アプリケーションを利用して、有用なデータと大きなビジネス価値を得ています。

多種多様な分野において ODS という共通のアプローチにより、財界人から物理学者、開発者、個人までもが利益を得ることができます。ODS と HPC を結び付ける新しいツールのおかげで、データによって世界を理解する進歩のペースはさらに速くなるでしょう。

HPC、ビッグデータ、高度な解析はもはや無関係ではありません。組織が未活用データの価値をコスト効率良く利用するには、オープン・データ・サイエンスを採用するべきです。オープンソース・テクノロジーは高速化のために構築されたものではありません。しかし、今日のビジネスでは、スピードが求められています。インテル® MKL により強化された Anaconda* のハイパフォーマンスな解析により、データ・サイエンス・チームは急激な変化にも簡単に対応することができるでしょう。

また、組織が独自に ODS エコシステムを管理している場合、組織の方針変更による影響を受けます。その組織に特有のシステムになりがちなため、内製のソフトウェア (以前の担当者がフリーソフトに手を加えて構築していることもあります) からオープンソース・ソフトウェアへのデータや人的リソースの移行は困難になります (特にハードウェアの変更を伴う場合)。データサイエンスのように革新的で個別に調整が必要な分野では、隠れたコストは多くのマネージャーが考えているよりも大きくなります。

これらの隠れたコストは、理想化されたゼロコストのデータ・サイエンス・アプローチでも、実際には想定以上のコストがかかることを意味します。徹底的にテストされ、完全に互換性がある、定期的に更新されるソフトウェアのエコシステムを提供する ODS ベンダーが、総保有コストの点で勝者となるでしょう。

ODS コミュニティーとデータサイエンスの未来

ODS は、革新や低コストにのみ取り組むのではなく、コミュニティーにも力を入れています。

Python* と ODS は、コミュニティーにより構築されたソフトウェアの優れたモデルです。プロプライエタリーなソリューションに対する利点により、オープンソース・ソフトウェアの利用は増加しています。毎年、新しい Python* 開発者が大学の教育課程から誕生しています。これらの開発者は、GitHub や StackOverflow のようなオンライン・ソフトウェア・フォーラムに参加して、オープンソース・ソフトウェアのパッチを提供したり、独自のオープンソース・プロジェクトを開始しています。ODS の明白な利点により、データ・サイエンティストやドメイン・エキスパートは Python* やそのパッケージスイート (NumPy*、pandas*、matplotlib、dask、scikit-learn、その他を含む) を学んでいます。

優れた ODS プラットフォームである Anaconda* は、データ・サイエンス・チームが、高価値で影響力のあるデータ・サイエンス・ソリューションの作成に集中できる、新しい時代への門戸を開きました。Anaconda* は、現在、そして成長し続けるスケーリング要件に簡単に対応することができるでしょう。

参考資料 (英語)

Numba* >

Anaconda* のサブスクリプション >

オープン・データ・サイエンス >

Anaconda* ホワイトペーパー : オープン・データ・サイエンスの旅 >