高速データ処理基盤 M^3 for Batch Processing

(Asakusa on M^3BP)

2016 年 4 月 12 日

目次

1 イントロダクション

M^3 for Batch Processing (以下 M^3 for BP) は DAG (Directed Acyclic Graph; 有向非巡回グラフ) の形で表現されたタスクをマルチコア環境で効率よく処理するためのフレームワークです。DAG ベースのタスク表現は Apache Spark などの分散処理基盤でも利用されていますが、M^3 for BP では単一ノードでの処理に特化することで、高速かつ高い費用対効果を実現します。

Asakusa Framework は業務システムのバッチに並列・分散処理の能力を活用するためのフレームワークです。Asakusa on M^{3}BP は、Asakusa DSL を始めとする Asakusa Framework の開発基盤を利用して作成したバッチアプリケーションに対して、M^3 for BP をその実行基盤として利用するための機能セットを提供します。Asakusa on M^{3}BP では実行基盤として M^3 for BP を使用することで、Asakusa Framework を用いて記述された Apache Hadoop や Spark 向けの業務バッチを、単一ノード上で高速に処理することができるようになります。とくに、小規模〜中規模のデータに対しては単一ノードでありながらも Spark よりも高い性能を実現することができました。

2 開発の背景

2.1 業務バッチ処理の現状

近年、Hadoop や Spark などの扱いやすい分散処理基盤の台頭もあり、数百ギガバイト・数テラバイトと大量のデータを分析し業務に活かすという事例がみられるようになってきました。Asakusa Framework はこれらの分散処理基盤を活用し、長い時間を要していた業務バッチ処理を高速に行うことを可能としてきました。

しかし、実際に運用されているシステムでは数ギガバイトから数十ギガバイト程度の小規模〜中規模程度のデータを処理できれば十分というケースも多く、そのようなワークロードを Hadoop や Spark で処理しようとすると、処理すべきデータ量に対してシステムが大掛かりになってしまったり、分散処理のためのオーバーヘッドが相対的に大きくなったりしてしまうという問題がありました。

従来の手法では時間がかかりすぎるが分散処理では大掛かりすぎる。そのようなワークロードをより効率よく処理できるようになることで、より多くのケースで最適なソリューションを提供することが可能となります。

2.2 計算ノードの高性能化

CPU のコア数の増加、メモリ容量の増加といったコンピューターハードウェアの進歩はまだ続いており、単一ノードで 70 個を超える CPU コアや、テラバイト級のメモリを搭載するシステムも登場しつつあります。このような高性能の計算ノードを用いることで、従来であれば複数ノードを用いて分散化しなければ処理できなかった規模のデータ処理を単一ノードのみで処理することが可能となりつつあります。

これまでクラスタシステムを用いて行っていた処理が単一ノードで行えるようになると、マシン台数を減らしたりそれに伴う複雑な運用が簡略化されたりするなど、運用コストの大幅な削減などの効果が期待されます。

3 特長

3.1 小規模～中規模データ処理に特化した性能

Hadoop MapReduce や Spark はそのスケーラビリティの高さから、数百ギガバイトやテラバイト超といった大規模データ処理において非常に高い性能を示します。一方で、数ギガ～数十ギガバイト程度の小規模～中規模データの処理においては性能が伸び悩んでしまうという問題がありました。この原因として、Hadoop や Spark は数千台の計算ノードからなるクラスター環境を想定して設計されていて、小規模～中規模データを取り扱う際には無駄が大きいという点や、複数の計算ノードにまたがってデータ交換を行うため、ネットワーク通信やストレージ入出力に余計な処理が必要になっている点が挙げられます。

M^3 for BP では、MapReduce や Spark が苦手とする小規模～中規模データ処理において、単一ノード上で処理を完結させます。これにより、計算ノード上の CPU コアやメモリの細かい制御が可能となり、さまざまな計算リソースを無駄なく最大限に活用することが可能となります。また、全ての処理を計算ノードのメモリ上で完結させるため、ネットワークやストレージ入出力など、データ交換のオーバーヘッドを大幅に削減できます。結果として、単一ノードあたりの処理効率を大幅に向上させることができ、小～中規模の単一ノード上で処理を完結できるような処理に対しては、複数の計算ノードを用いた分散処理よりも高速な処理を実現できる場合があります。また、より多くのデータを単一ノードで処理することが可能となるため、バッチ処理に使用するシステムの小型化も期待することができ、高い費用対効果を実現しています。

M^3 for BP のアーキテクチャ

M^3 for BP は図 1 のようにアプリケーションなどの上位層から与えられた DAG に対して、シャッフル処理（データのキーによる分割やソート）などを行うタスクを追加したり、分割してより細かい粒度のサブタスクの集合に変換したりすることによってタスクを並列処理しやすい形に変換しています。

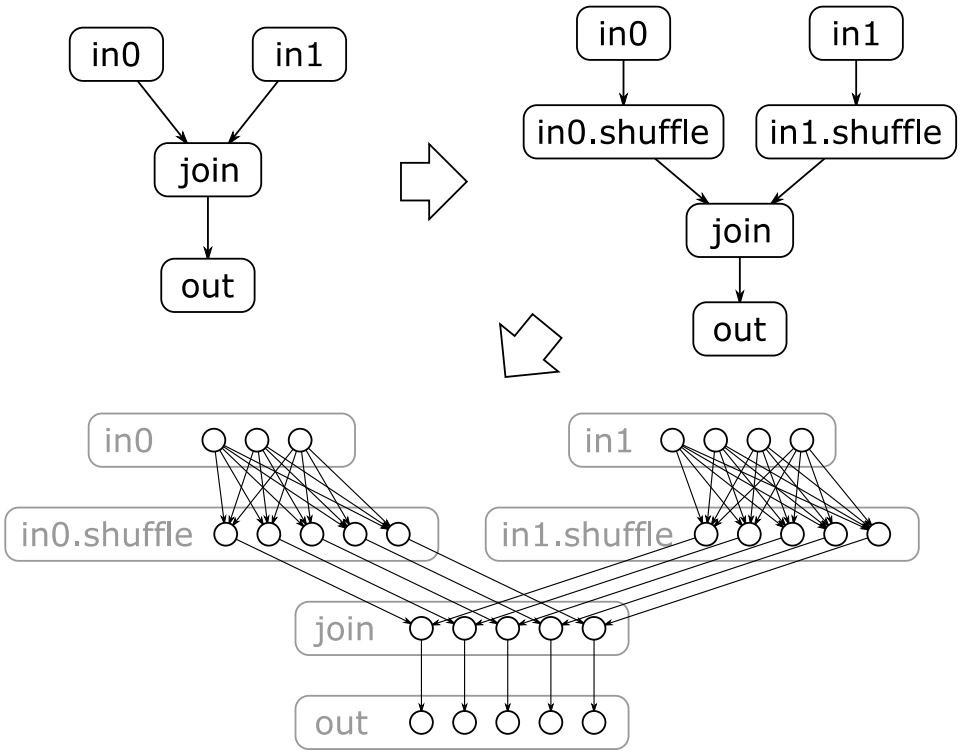

図 1　グラフ変換の例

Spark などの分散実行基盤でも同様の変換が行われていますが、これらは大規模なクラスター上での分散処理を前提としているため、タスクを分解する粒度を細かくしすぎると、逆に通信などのオーバーヘッドの分だけ速度が低下してしまうという問題があります。しかし、タスクを分解する粒度が粗い場合には、各 CPU コアに割り当てるタスクの大きさにばらつきが出てしまい、結果として計算ノード上の各コアに均等にタスクを割り当てることがむずかしく、CPU を有効に活用できないという問題があります。特に、小規模〜中規模のデータではそもそものタスクが小さくなりがちで、この傾向が顕著です。

対して M^3 for BP では単一ノードに閉じていて通信などのオーバーヘッドが非常に小さいため、非常に細かな粒度のタスクに分解することが可能です。また、タスクが終了して CPU コアに空きができたとしても、即座に次のタスクをその CPU コア上で実行させることができます。これらにより、大規模な分散システムに比べて、マルチコアを有効に活用しシステム利用率を高めることができます。さらに、単一ノードのメモリ上ですべての計算を行っているため、分散システムでコストのかかるタスク間のデータ交換等が発生せず、高速な処理を実現しています。

3.2 処理系可搬性

Asakusa Framework を用いて作成されたアプリケーションは、ビルド時のオプション設定のみでアプリケーションコードに手を加えることなくバックエンドを変更することができます。これまで Asakusa Framework がサポートしていた処理基盤は Hadoop MapReduce (Asakusa on MapReduce)と、Spark (Asakusa on Spark)の二つでしたが、今回新たに M^3 for BP が処理基盤に追加されました。これらはいずれも同一のアプリケーションコードで動作するため、小規模から大規模のバッチまで同じプログラムで効率的に動作させることができるようになります。

システム運用開始時には M^3 for BP を用いて費用対効果の高い小規模なシステムを構築し、運用が進むにつれて単一ノードで処理できなくなったら大規模なデータに強い Spark に移行するようなスケーラブルな運用が可能になります。

3.3 Java との共存

Asakusa Framework は Java を利用してアプリケーションを開発するフレームワークであり、Hadoop や Spark も Java 仮想マシン（以下、JVM）の上で動作するプラットフォームです。Java はプログラムの安全性や生産性が比較的高いというメリットがある反面で、JVM が介在することにより多数の CPU コアや大容量のメモリを十分に活用するのが難しいという問題もあります。

上記の問題に対処するために、M^3 for BP は C++を利用したネイティブ（JVM を利用しない）ライブラリとして作成しています。これと組み合わせた Asakusa on M^3BP では、CPU コアの制御やメモリの管理などを M^3 for BP に任せてしまい、安全性や生産性が重視される業務アプリケーションの部分だけを JVM の上で動作させています。これらの組み合わせにより、Asakusa Framework を利用して作成されたアプリケーションの可搬性を担保し、安全性や生産性を保ちながらも、高い処理性能を実現しています。

4 性能評価

Asakusa Framework を利用して作成された実アプリケーションにおいても M^3 for BP は良好な性能を示しています。この章では、実際に業務システムとして使用されているバッチアプリケーションを用いて従来の分散処理基盤を用いた場合と Asakusa on M^{3}BP の性能を比較します。

4.1 ワークロード概要

対象となる処理は、ある食品製造業で実際に行われている原価計算のバッチ処理です。はじめに、製品と原材料のマスターデータ（約 80 万件）から、製品毎の BOM（約 200 万件）へ展開（図 2）して計算処理を行い、製品ごとの原価計算を行います。

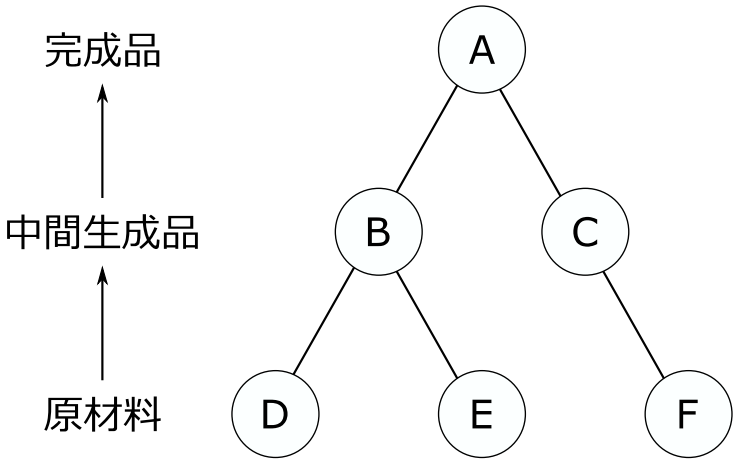

図 2　BOM の展開の例

次に、この求めた製品原価を、生産と出荷ごとにコストを整理して計算を実施します。生産ごとのコストとは、製造している場所、すなわち、工場・工場のラインなどでのコストを指します。出荷ごとのコストとは、得意先ごと、すなわち販売先ごとのコストを指します。ここまでの計算を行った上で、計算結果を工場のマネージャーや現場の担当者が確認しています。

このバッチ処理は、1 日 2 回実施されており、1 回あたりの処理での入力データサイズは 4〜5GB です。

4.2 評価環境

性能評価には Microsoft Azure 上に構築された環境を使用しました。インスタンス情報や使用したソフトウェアのバージョンを以下の表に示します。なお、分散環境の計算

用ノード数は Spark を用いたときに最もノードあたりの効率が良くなるように選択されています。

	MapReduce	Spark	M³ for BP
バージョン	2.7.2	1.6.1	0.1.0
Java 処理系	Java SE Development Kit 8 Update 74		
C++コンパイラ	N/A		GCC 4.8.5
OS	CentOS 7.1		
インスタンスタイプ	Microsoft Azure Virtual Machines D5 v2 (16 CPU コア・メモリ 56GB)		
計算用ノード数	5 (*)		1

*: 計算用ノードとは別に管理用ノードが 1 台必要

4.3 評価結果

各評価環境でアプリケーションを実行し、その際に要した時間を図 3 に示します。

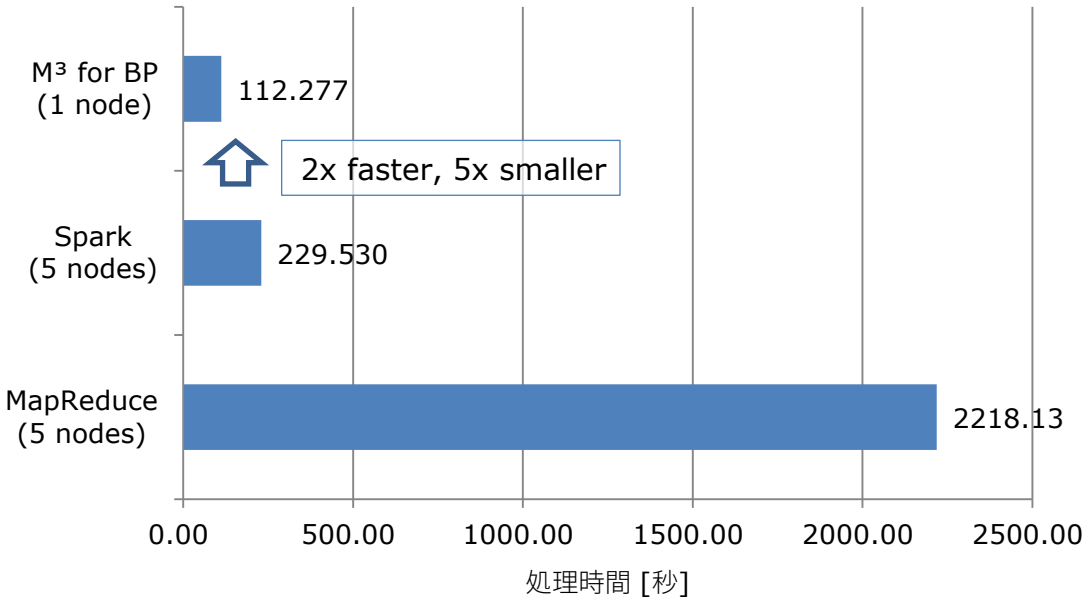

図 3　性能評価結果

このように Asakusa on M³BP を使用することにより、実アプリケーションにおいて Asakusa on Spark を用いた場合に比べて 1/5 のノード数で約 2 倍の性能、合わせて約 10 倍の費用対効果を達成しています。

5 適用領域

5.1 Asakusa Framework の適用領域

2.1 で触れているとおり、Asakusa Framework と Hadoop や Spark の組み合わせで、数百ギガバイト以上の業務に関わるデータ処理を高速化してきました。主に、ホストやメインフレーム、あるいは、データベースで長時間をかけて実施していたバッチ処理に類するようなものです。具体的には、以下のような処理を対象としてきました。

- 業務バッチ処理での利用
  - 夜間や数日をかけて行われる業務データのバッチ処理。
  - 会計の締め処理・原価計算・一括でのコード変換といったデータ処理。
- 分析データ加工での利用
  - 分析システムで、複数の業務システムから分析用のデータベースへデータを投入する際に行われる変換や集計などの処理を高速化。
  - データ分析時にデータベース側で集計などの加工処理を極限までおさえて、閲覧のスピードも高速化。
- シミュレーションデータの作成
  - 意思決定を行うために、検討したい情報が複数のパターンがあるケース（売上の数字・仕入品の価格の変動、など）で、全てのパターンを一度に計算を行う。

5.2 Asakusa on M^{3}BP の適用領域

Asakusa on M^{3}BP では、これまでの Asakusa Framework が対象としていた業務データ処理の高速化を行います。ただし、Hadoop や Spark とは大きく異なる単一ノードという特長を活かし、以下のような活用を行うことができます。

- 単一ノードで並列分散処理を始める
  - Hadoop や Spark の導入で、たくさんのノードにコストをかけられないことが多くあります。サーバーの購入だけでなく、どこに設置するか、運用はどうするかといった、環境や運用の面での整備も求められます。
  - M^3 for BP は、単ノードでの導入となるため、複数ノードを持つことに関する課題は解消されます。

- データ量が増えていった時には、3.2 の可搬性に記述してあるとおり、Hadoop や Spark の環境を構築すれば、コードを変えることなくアプリケーションの移行を実施することができます。

- 拠点毎にデータ活用を行う
  - Hadoop や Spark を用いた並列分散処理環境は、最低 5 台のクラスター構成を組む必要があるため、設備の整った環境で構築をします。
  - 複数拠点を持つ（国内/国外の生産拠点・営業拠点・店舗など）場合は、本部へは全てのデータを送ることができずに、拠点単位でデータを収集、分析し、拠点の集計結果のみを送って連携していることが多くあります。
  - M^3 for BP は、単ノードで並列分散処理環境を構築できるので、拠点毎にサーバーを設置し、拠点毎にデータを集約、本部で必要としているデータに加工することも可能です。