特集論文

大規模データ分散処理技術を活用したバッチ高速化ソリューション

中島賢弘* 佐藤彰洋*
武石富士見*
佐藤啓紀*

Batch Speeding−up Solution by Distributed Processing Technologies

Takahiro Nakajima, Fujimi Takeishi, Hiroki Satou, Akihiro Satou

要　旨

バッチ処理は，多くの企業の基幹業務システムで現在でも重要な役割を担い，取り扱うデータ量は増え続けている。データ量増加に伴いバッチ処理時間は増大し，処理時間の短縮は企業が抱える重要な課題の１つとなっている。

三菱電機インフォメーションシステムズ㈱(MDIS)では，大規模データの分散処理を支えるフレームワークであるHadoop(注1)技術に着目し，この課題を解決するための“バッチ高速化ソリューション”を立ち上げた。

このソリューションのサービス構成は次のとおりである。

⑴　高速化診断サービス

現行バッチ処理のボトルネック箇所を診断し，このソリューション適用による改善効果を提示する。

⑵　高速化設計・構築サービス

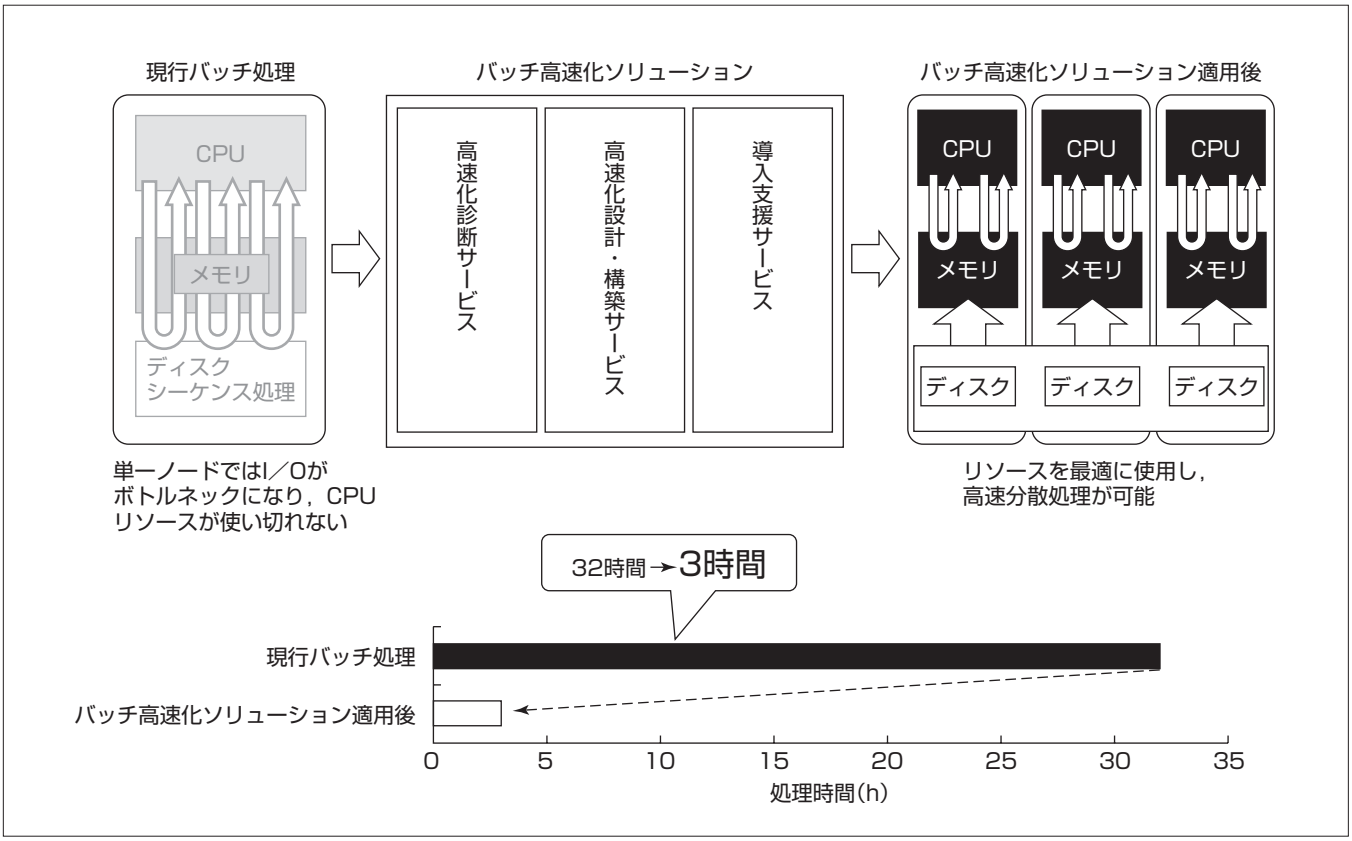
診断結果を基に現行バッチ処理を詳細分析し，高速化処理設計と高速化モジュール構築を行う。

⑶　導入支援サービス

高速処理環境の構築と稼働までの支援を行う。

このソリューションを，年次バッチ処理の時間短縮が長年の課題であった顧客システムに適用し，ディスク，CPU(Central Processing Unit)，メモリなどのリソース全体の効率化，及び分散処理化などを行うことで，現行32時間掛かっていた処理を３時間に短縮し，約1/10の大幅な性能改善を達成した。

（注１）　Hadoopは，The Apache Software Foundationの登録商標である。



バッチ高速化ソリューションと適用事例

ディスクI／O（Input／Output）ネックが原因で処理時間に課題を抱えているバッチ処理に対し，バッチ高速化ソリューションを適用することで，ディスク，CPU，メモリなどのリソース全体の効率化及び処理の分散化によって処理時間短縮を実現した。顧客システムへ適用した事例では，32時間の処理を3時間に短縮することができた。

*三菱電機インフォメーションシステムズ㈱

1. まえがき

バッチ処理の多くは，長年にわたり企業の基幹業務システムを支えている。業務を支援するシステム化範囲の拡大や扱うデータ量の増加に伴い処理時間が増大し，夜間日次バッチ処理がオンラインサービス開始時間までに終わらない等，企業にとって大きな課題となっている。

MDISではこの課題を解決するために，近年大規模データの分散処理を支えるソフトウェアフレームワークとして注目され，企業への導入が進むHadoop技術に着目し“バッチ高速化ソリューション”を立ち上げた。

本稿では，このソリューションの概要，バッチ処理に課題を抱える顧客システムへの適用事例と適用ポイント，及び適用後の効果について述べる。

2. バッチ高速化ソリューションの概要

このソリューションでは，①高速化診断サービス，②高速化設計・構築サービス，③導入支援サービスを提供している。処理性能に問題がある現行バッチ処理に対し，このソリューションを適用することで，顧客に手間をかけることなくバッチ処理の高速化を実現する。

2.1 高速化診断サービス

高速化診断サービスでは，このソリューション適用による改善効果を次の2段階で提示する。

2.1.1 簡易診断サービス

対象データ量，処理概要から簡易的な改善効果有無を無償で診断する。

2.1.2 詳細診断サービス

対象データ量，処理概要，現行処理時間，現行リソース(設計書やプログラムソースなど)の情報を基に詳細診断を行う。対象バッチ処理のボトルネック箇所の解析を行い，このソリューション独自の“見込み処理時間計算式”で高速化後の見込み処理時間を算出し，改善効果を提示する。なお，改善効果については，ハードウェア構成パターン(サーバスペック，ノード数)別に導入に必要なハードウェア概算費用も併せて提示する。

2.2 高速化設計・構築サービス

高速化設計・構築サービスは，高速化診断サービスでインプットにした情報を基に，処理対象データの流れに着目した高速化設計と高速化モジュール構築を行うサービスである。

高速化モジュールを構成するソースコードの一部は，高速化設計書から自動生成可能である。また，現行資産を活用した構築を実現しているため，仕様検討など顧客に手間をかける時間を，通常の新規構築の場合と比較して大幅に削減することができる。

2.3 導入支援サービス

導入支援サービスでは，Hadoop環境の構築，高速化設計・構築サービスで開発したモジュールの導入，稼働までの支援を行う。

3. バッチ高速化事例と適用ポイント

このソリューションを適用した顧客では，年度始めに処理時間の長い年次バッチ処理を実施するため，土日(休日)から月曜日(営業日)にまたがって基幹業務システムのオンラインサービスを停止しなければならず，通常業務に支障をきたしていた。

このソリューションの適用によって，年次バッチ処理(全体40時間処理)のうち，特に時間がかかっていた32時間のバッチ処理を3時間に短縮することができた。

次に，顧客に対し提供した高速化診断サービス(詳細診断サービス)，高速化設計・構築サービスの具体的な内容，適用に当たって工夫したポイント，及び高速化後の性能の結果について述べる。

3.1 高速化診断事例と適用ポイント

3.1.1 高速化処理対象選定

ボトルネック箇所の高速化が可能かどうかを見極めるため，次に述べる2段階のアプローチによって高速化すべき対象を絞り込んだ。

1段階目：各処理の処理時間測定による高速化候補の選定

2段階目：高速化候補と高速化しない現行処理間の結合度調査

1段階目で各処理の処理時間の測定結果に基づき，処理時間がかかるものから高速化処理対象の候補とした。Hadoop環境上でバッチ処理を実施する場合，現行環境(データベースサーバ)とHadoopサーバ間のデータ転送時間がオーバーヘッドとなる。そのため，現行処理時間が短い処理は，このオーバーヘッド時間を考慮すると高速化の効果が見込めない可能性が高いので高速化対象としない。

このソリューションでは高速化効果が見込まれるしきい値(処理時間)を設定しており，しきい値による分類で年次バッチ処理の全250処理中25処理を高速化対象候補とした。この25処理の合計処理時間は，全処理時間40時間中32時間(約82%)を占めており，有効な高速化候補であることが分かる。

次に2段階目では，高速化候補の25処理と高速化対象外の225処理の結合度を調査した。結合度とは，処理間の依存関係を示す度合いであり，例えば処理Aの結果が処理Bの処理内容に影響を及ぼす場合は結合度が高いと言える。結合度を調査した理由は，高速化処理環境と現行処理環境は異なる環境での処理であり，処理間の結合度が高い場合には，高速化による効果が期待できないためである。

調査した結果，高速化候補の25処理と高速化対象外の225処理で，結合度は低いことが判明したため，25処理を高速化対象として確定した。この適用例では，結合度が高

い高速化対象はなかったが，顧客のバッチ処理の全体像を把握した上で，単純に時間のかかる処理を選定するのではなく，現行処理との結合度を見極めた上で対象を選定することが重要である。

3.1.2 高速化後性能の見極め

選定した25処理に対し，このソリューション独自の“見込み処理時間計算式(注2)”を用いて，バッチ処理高速化後の見込み処理時間を算出した。その結果，25処理の見込み処理時間は高速化前の32時間に対して高速化後は5.6時間との結果になり，適用によって処理時間に相当の短縮が見込めると判断した(**図1**)。

(注2) このソリューション立ち上げに当たり，バッチ高速化のプロトタイプを作成し，インプットデータ量，処理内容を基に複数パターンについて検証して処理時間を実測した。実測結果から基礎数値(データ量単位の処理時間)を求め，計算式を策定している。

3.2 高速化設計・構築事例と適用ポイント

高速化バッチモジュールの処理は，現行バッチ処理が直接データベースに処理を実施する(**図2**(a))のに対し，**図2**(b)に示すとおり①データ抽出処理(データベース→Hadoop(HDFS(Hadoop Distributed File System)))，②変換処理(Hadoop処理)，③データ反映処理(Hadoop(HDFS)→データベース)の3ステップ処理となる。

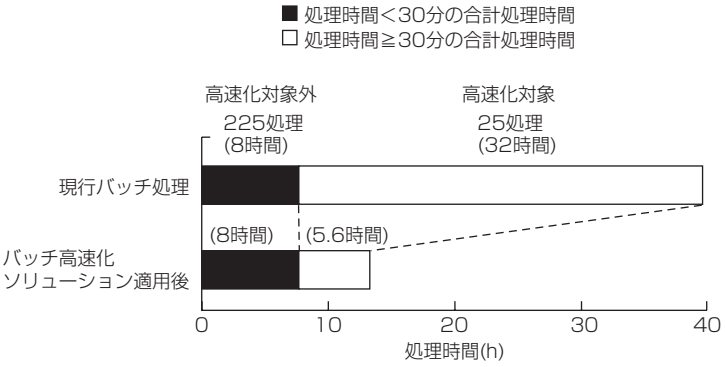


図1．バッチ処理時間の比較

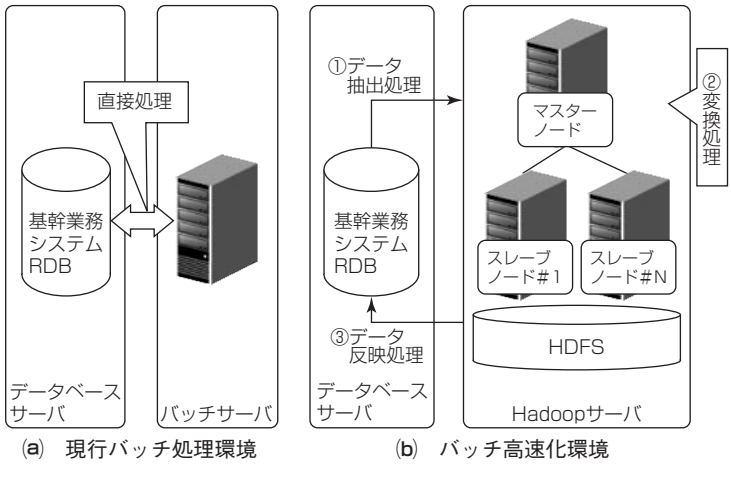


図2．バッチ高速化構成イメージ

次に，これら3つの構築ポイントについて述べる。

3.2.1 データ抽出処理とデータ反映処理の最適化

先に述べたとおり，データ抽出処理とデータ反映処理は，Hadoop環境で処理する場合にはオーバーヘッド時間となる。特にデータ反映処理は既存のデータベースサーバ環境の書き込み性能に大きく依存する。このソリューションでは，このオーバーヘッド時間を最小化する①データベースサーバとHadoopサーバ間の転送量削減，②抽出・反映方式の最適化の2つの工夫を行っている。

①の転送量削減については，処理対象テーブルの中に処理に必要な項目と不要な項目があることに着目し，抽出・反映に必要な項目のみを対象とすることで実現している。

②の抽出・反映方式の最適化については，幾つかの方式を数種類のデータパターンごとに実測し，最適な抽出・反映方式を選定することで実現している。

抽出方式は，方式1：sqoop(注3)，方式2：SQL(Structured Query Language)*Plus(注4)，方式3：WindGate(注5)を候補として，各方式による抽出時間を測定し，どのデータパターンでも最速であった方式1を今回の事例では採用した。

一方，反映方式は，方式1：sqoop，方式2：PL(Procedural Language)/SQL(注6)，方式3：SQL*Loader(注7)を候補として，反映時間を測定し最適手法を選定した。反映方式の場合は，処理対象テーブルの全件数に対する更新件数の比率によって最適手法は異なり，この更新件数比率をしきい値として，手法を選定している。更新件数がしきい値未満の場合は方式2が最速であり，しきい値以上の場合は方式3が最速である。このソリューションの反映方式ではデータ特性によって方式を切り分けて対応している。今回の事例では更新件数の割合がしきい値未満であったため，方式2を採用している。

さらに，抽出・反映処理については構築時の生産性向上をねらい，高速化設計書を読み取って処理を実行するソースコードを自動生成する機能を実現している。

(注3) データベースとHadoopの間でデータ転送を行うためのコマンドラインインタフェースアプリケーションである。
(注4) Oracle社によるデータベース操作のためのユーティリティである。
(注5) ノーチラス・テクノロジーズ社が開発したデータベースと連携するコンポーネントである。
(注6) Oracle社がデータベース言語SQLを独自に拡張したプログラミング言語である。
(注7) 外部ファイルのデータをOracle Databaseの表に取り込むユーティリティである。

3.2.2 変換処理の実現方式について

変換処理の高速化バッチモジュール構築には，ノーチラス・テクノロジーズ社が提供するOSS(Open Source Software)“Asakusa Framework(注8)(注9)(以下“AsakusaFW”という。)”を活用している。高速

化モジュールを構築する場合，分散処理を意識したプログラミングが必要となるが，AsakusaFWは分散処理を意識することなくプログラミングが可能であり，高速化モジュールを自動生成することができる。ただし，AsakusaFWによるプログラムは現行のバッチプログラムと構造やコーディング方法が大きく異なるため，現行のバッチプログラムを単純に機械変換して作成できるわけではない。そのため，このソリューションでは，**図3**に示すとおり，開発プロセスの中で次の3つの手段によって，バッチ処理高速化環境構築をする際の顧客への負担を少なくした。

⑴　現行バッチ処理内容の解析作業の効率化

このソリューションでは，独自の解析シートを用いて現行のバッチプログラムを解析する。解析シートは，高速化設計移行時に，現行のバッチプログラムの業務仕様を漏れなく移行すること，及び業務仕様に関係がない処理は排除することを目的として，必要な処理と不要な処理の分別の指標を示している。例えば，必要な処理はSQLなどのデータを加工する処理であり，不要な処理は変数に値を格納するような処理である。この解析シートを用いることで解析作業の効率化を図っている。解析作業はこのように既存リソースを主体としているため，顧客から現行仕様をヒアリングする必要がなく顧客の負担，変換コストを削減している。

解析結果は，“現行バッチ処理データフロー図”“プロセス処理内容”となる。

⑵　データフロー図を用いた高速化設計

AsakusaFWではデータフロー制御，データ操作，データ結合など処理特性に合わせた“演算子”が用意されており，演算子を組み合わせることでバッチ処理を構築する。

このソリューションでは，⑴のアウトプットの“現行バッチ処理データフロー図”“プロセス処理内容”をインプットとして，演算子を最適に組み合わせることで“高速化バッチ処理データフロー”を作成する。**図4**に現行バッチ処理のデータフロー図と高速化バッチ処理データフロー図の作成イメージを示す。移行設計後(**図4**の右側)は，データベースの内容はデータモデルに格納して取り扱い，業務処理はAsakusaFWの演算子に置き換えを実現する設計としている。また，各演算子に具体的な処理内容を定義し，現行処理内容の高速化設計を行う。設計には高速化設計書を用意しており，AsakusaFWのプログラム構造に準拠した定形フォーマットとなっている。

⑶　ソースコードの一部自動生成による構築作業の効率化

高速化モジュールのソースコードについては，高速化設計書から一部自動生成可能であり，開発コストを抑える工夫をしている。高速化設計書は先に述べたとおり，AsakusaFWの構成に準拠した構成となっているため，効率的な構築を可能にしている。

(注8)　Asakusa Frameworkは，Hadoop上で大規模な基幹バッチ処理を行うためのフレームワークである。
(注9)　Asakusa Frameworkは，㈱ノーチラス・テクノロジーズの登録商標である。

3.2.3　ジョブ構成の最適化

バッチ高速化ジョブは，**3.2節**で述べたとおり①データ抽出処理，②変換処理，③データ反映処理の3ステップの処理で成り立つ。ジョブ構成は**図5**のように段階的に各処理を実行することでデータベースサーバ，又はHadoopサーバに負荷が集中しないように工夫している。

3.3　高速化後の性能の結果について

3.1.2項で述べたとおり，高速化診断サービス段階では現行32時間の処理は5.6時間に削減可能であると見込んでいた。

これに対して，実際にはさらに，**3.2.1項**で述べた抽出・反映処理の転送量の削減，**3.2.3項**で述べたジョブ構

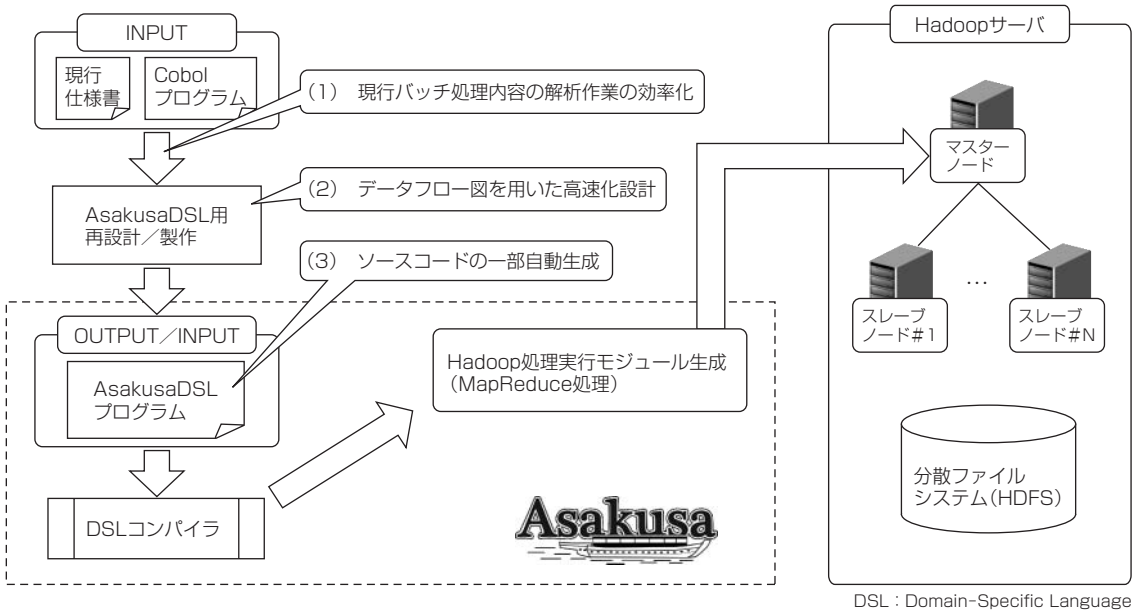
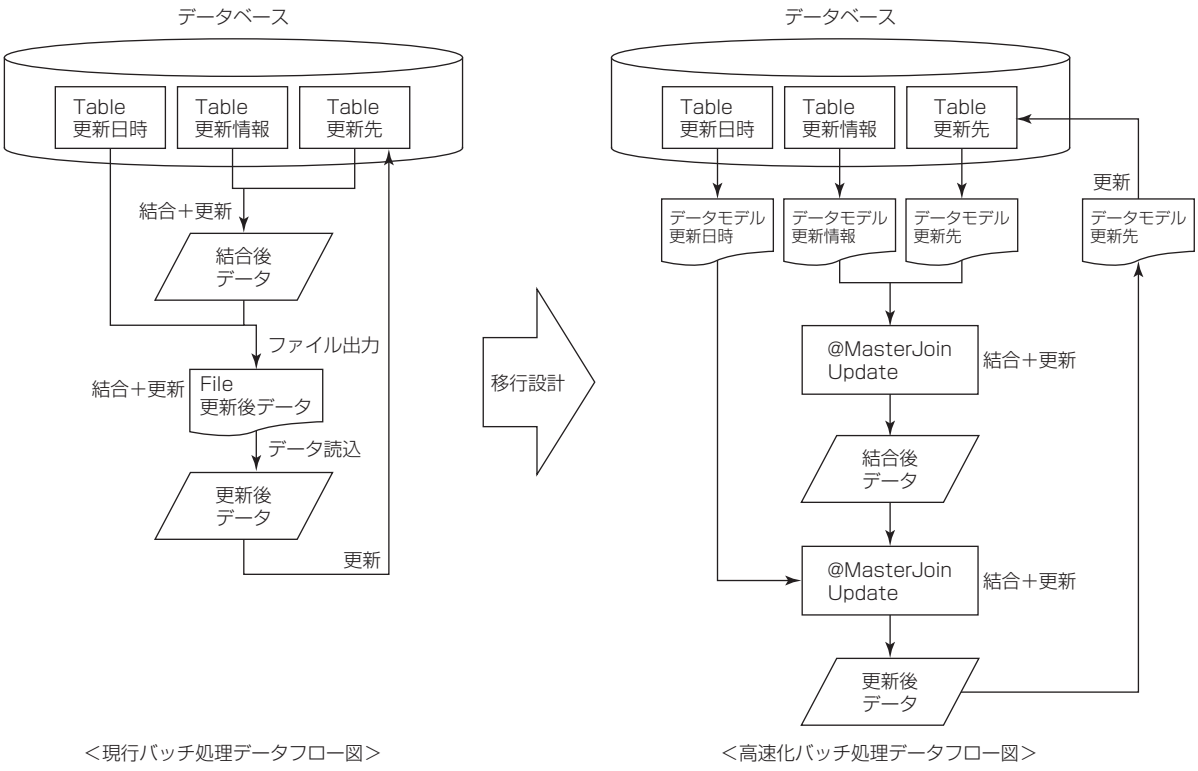


図3．変換処理の実現方式



＜現行バッチ処理データフロー図＞　　＜高速化バッチ処理データフロー図＞

図４．データフロー図の作成イメージ

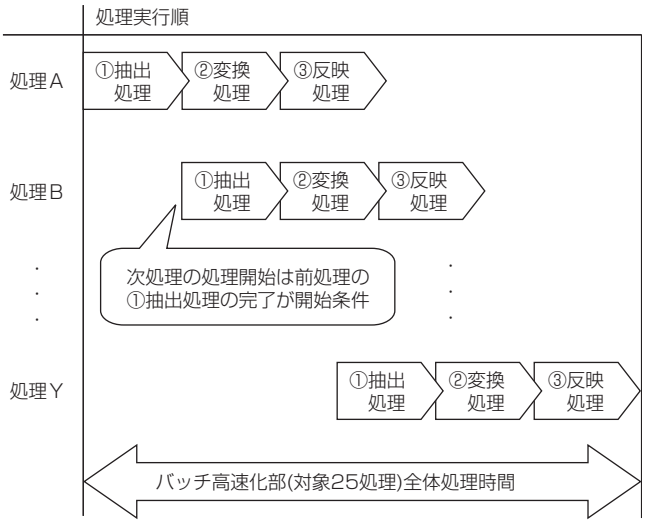


図５．高速化処理の実行順イメージ

成の最適化を実施する等，設計フェーズ，構築フェーズでも高速化するための様々な工夫を行うことによって，構築後の実測値では処理時間を３時間にまで短縮することができた。

4. む　す　び

今後，このソリューションの有効性を更に高めるためには，高速化設計・構築の自動化範囲の拡大，データベースとHadoop間の転送処理を高速化する必要がある。

また，現行バッチ処理を高速化することによって，月次集計しかできなかった業務データの日次集計，リアルタイム集計が可能になり，経営的判断を行う上で必要なデータをタイムリーに提供できるようになる。顧客にバッチ高速化ソリューションを導入することによってどのような効果や付加価値をもたらすことができるか，研究を更に深めてこのソリューションの価値向上に努めていく所存である。

一方，このソリューションで培った“高速にデータ加工するノウハウ”は，今後ビッグデータ分析を行う前処理であるデータクレンジング等への応用が可能であり，またHadoop環境はデータ分析基盤への活用が可能である。このソリューションを応用していくことで，適用範囲の更なる拡大・展開を進めていく。