「マルチメディア，分散，協調とモバイル
(DICOMO2018)シンポジウム」 平成30年7月

高速性と柔軟性を兼ね備えた分析プラットフォームの提案

杉本 健[1] 中田 侑[1] 郡浦 宏明[1] 木下 雅文[1]

コスト削減や売り上げ拡大、新規サービスの創出を行うため、価値のあるデータを膨大なデータから抽出する、データ分析が広く注目されている。データ分析では、複数の分析ツールを組み合わせた分析システムを構築し、膨大なデータの分析を行った上で、分析結果を元した仮説検証を何度も繰り返し実施する。そのため、膨大なデータを素早く処理する高速性と、複数の分析ツールを低い工数で組み合わせられる柔軟性が求められる。本論文では、データ分析をシーケンスとして定義し、シーケンスから低負荷で分析ツールを並列に実行可能にするプラットフォームを提供すると共に、シーケンス再設計を容易化することで、高速性と柔軟性を兼ね備えた分析プラットフォームを提案する。本プラットフォームを活用して分析システムを実装した結果、サーバ数に応じて分析プログラムのスループットがスケールし、提案するプラットフォームの負荷が低いこと、また、分析シーケンスの再設計時の実装工数が 4 日と十分低いことを確認した。本プラットフォームを用いることで、様々なデータ分析の結果をより素早く得ることが可能となり、さらなるコスト削減や売り上げ拡大、新規サービスの創出に貢献できると考えている。

Proposal of High Performance and Flexible Analytic Platform

KEN SUGIMOTO[1] YU NAKATA[1]
HIROAKI KONOURA[1] MASAFUMI KINOSHITA[1]

1. はじめに

近年、分析技術の発展により、実世界から得られた膨大なデータから価値ある情報を抽出し、コスト削減や売上拡大、新規サービス立ち上げなどの新たな価値を創出する取り組みが広く注目されている。例えば、レストランにおける店員の動きのデータと売上の POS データを分析することで、売り上げ向上に向けた店員の動きを検討する取り組み[1]や、インターネットショッピングサイトの購入履歴より、購買率向上に関わる要因の分析を検討する取り組み[2][3]が行われている。

このように大量のデータから新たな価値を生み出すためには、まず仮説を構築し、次に仮説に基づいてデータ分析を行い、最後に構築した仮説が正しいかを検証する、仮説検証のプロセスを何度も繰り返す必要がある。この仮説検証のプロセスの施行回数を増やすために、大量のデータを高速に分析するプラットフォームが求められている。

これに対し、データ分析を並列に実行することで高速に実行するツールが提供されている。たとえば、Apache SPARK®[4][5]を利用すると、プログラム実装者は Apache SPARK®に準拠したプログラムを実装することで、並列実行について意識せずに並列実行可能なプログラムを実装することが可能である。

一方、データ分析システムは、前処理や分析処理、可視化などから構成され、処理ごとに得意な分析ツールが必要なことや、過去に実装したプログラムの活用の観点から、複数のツールを組み合わせるとシステム構築が容易となる事が多い。そのため、分析ツールを１つに決めてしまうと、仮説検証ごとに必要となる分析システムの組み換えが難しくなり、分析システムの組み換えを低工数で実現する柔軟性が課題となる。例えば、前述の Apache SPARK®のような１つの分析ツールのみでシステムを組むと、分析ツールの組み換えが難しい。一方、利用したい分析ツールや、過去に実装済みのプログラムは並列実行に対応していないことが多く、そのまま利用した場合は実行速度が課題となる。

以上より、並列実行による高速性と、様々な分析ツールを簡単に組み合わせることを可能にする柔軟性を兼ね備えた分析システムの構築手法が求められている。

これに対し、様々な分析ツールを並列に実行するための機能を備え、分析システムをシーケンスとして定義する分析プラットフォームを提案する。本分析プラットフォームでは、分析をシーケンスとして定義し、シーケンスから呼び出す分析ツールを用いた分析プログラムを別途実装することで、分析ツールを簡単に組み合わせることを実現する。また、各サーバ上に分析プログラム実行基盤を導入し、分析プログラム実行基盤経由で分析シーケンスに合わせて分析ツールを並列に実行することで、低いオーバーヘッドで並列実行を実現し、高速性を確保する。

本報の以降の構成は以下の通りである。2 章では、データ分析システムについて述べ、高速性と柔軟性の必要性について明らかにする。3 章では、高速性と柔軟性を両立する分析プラットフォームを提案する。4 章では提案した分析プラットフォームの評価を行う。5 章ではまとめと今後の方針について述べる。

1（株）日立製作所

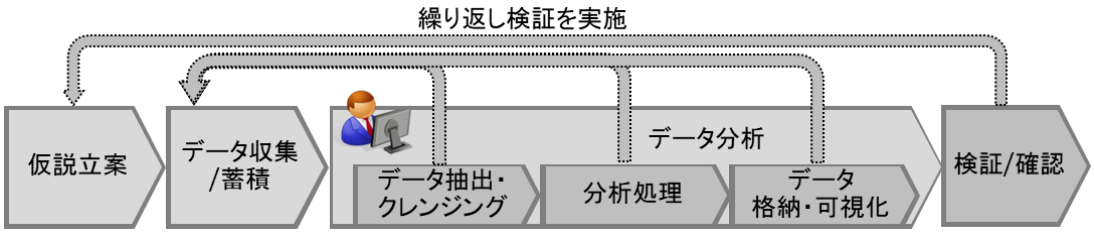

図 1 データ分析の全体フロー

2. データ分析の概要

2.1 全体フロー

データ分析実施時の全体フローを図 1 に示す。データ分析実施時はまず、データ分析によって検証を行いたい仮説を立案する。例えば、店員の動きに応じて店舗の売上が変化する、などの仮説を立案する。次に、仮説の検証のために必要となるデータの収集を行う。収集するデータとしては、装置や人に取り付けたセンサーのデータや、店舗の売上の POS データ、カメラの画像データ、気象データ・地図データなどのオープンデータ等が考えられる。前述の例では、店員につけたセンサーデータや、店舗内のカメラのデータ、そして店舗の売上の POS データなどが必要となると考えられる。データが揃ったら次に、仮説検証を行うために必要なデータ分析の処理の流れを決め、分析ツールを用た分析システムの構築を行う。データ分析では、次のような処理の流れになることが多い。まず、データ抽出及びデータクレンジング[6]などの前処理を行う。これは、集めたデータを分析ツールが利用しやすい形式に変更する作業である。例えば、分析に必要な部分のデータの抽出や、データ間で時間などの形式が異なる場合の形式の統一化などを行う。次に、分析ツールを用いた機械学習などの分析処理を行う。そして、分析処理の実行結果を格納し、可視化を行う。データ分析が終わったら、分析結果を元に、仮説の検証を行う。検証結果に応じて、仮説そのものの見直し、分析対象のデータの範囲変更や種別変更、アルゴリズムや分析ツールの見直しなどを行い、必要に応じて仮説再構築、データ再収集、プログラム再実装やシステム再構築の上、再度データ分析を実施する。例えば、店員の動きと売上の相関が弱いため、更に地理的なデータや曜日のデータを収集し、組み合わせて分析を行う、といったことが考えられる。

2.2 課題

分析対象のデータ量はデータ分析のケースそれぞれによって異なり、仮説検証毎に増減するが、一般的に数 GB から多い場合は数 TB 以上に上る。そのため、単純に分析ツールを実行した場合は分析時間が数日に及ぶことも有り、高速化の為の取り組みが必要とされている。

一方、データ分析は、複数の分析ツールを組み合わせてシステムの構築が行われることが多い。これは、次の理由による。まず、データ分析に必要な機能を全てカバー可能なツールは存在しない。前述のように、データ分析では、データのクレンジング[6]などの前処理や機械学習によるデータ分析、可視化など、複数の処理で構成される。それぞれ処理に向いたツールは異なるため、各分野の処理が得意なツールを組み合わせることが必要である。また、過去に開発されたプログラムの使いまわしや、開発者が使い慣れた分析ツールを活用することで、開発期間を短縮することが可能であり、プログラム流用や使い慣れた分析ツールの活用の結果として、様々なプログラムが分析システム内で利用される。

既に述べたように、分析システムでは分析毎に対象データの見直しや分析ツールの見直しが何度も行われるため、それに合わせて分析システムの構成や、分析システムを構成する分析ツールが変化する。そのため、必要に応じて利用する分析ツールを柔軟に入れ替え、システムを低い工数で変更できる柔軟性が、分析システムに求められている。

以上より、分析ツールを高速に実行する機能と、様々な分析ツールを柔軟に組み合わせられる機能を両立する、分析システム向けプラットフォームが求められている。

3. 分析プラットフォームの提案

本章では、高速性と柔軟性を備える分析プラットフォームを提案する。本プラットフォームを用いることで、様々な分析ツールを柔軟に組み合わせつつ、各ツールを高速に実行することを可能にする。本章ではまず、提案する分析プラットフォームの着眼点について述べる。次に、提案する分析プラットフォームの全体構成について述べる。そして、本プラットフォームの動作シーケンスについて述べ、その後に、各分析ツール実行サーバの管理方式について述べる。最後に、分析プログラム間のデータ受け渡し方式について述べる。

3.1 着眼点

様々な分析ツールを柔軟に組み合わせて利用するためには、分析ツールを呼び出す分析全体のシーケンスを管理し、分析施行毎にシーケンスを変更する方式が考えられる。ここで、分析シーケンスから呼び出す分析ツール自体は並列実行の機能を持たない場合が多い。これに対し、並列実行が必要な処理は、ロードバランサを用い、ロードバランサ経由で複数のサーバ間で処理を割り振ることが考えられる。しかしこの場合、ロードバランサの導入やロードバランサ自体の処理が必要となり、実装工数や処理負荷の増大

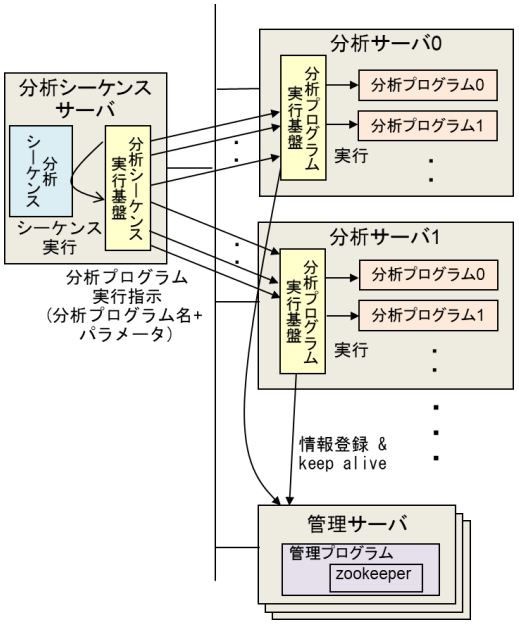

図 2 分析プラットフォーム全体構成

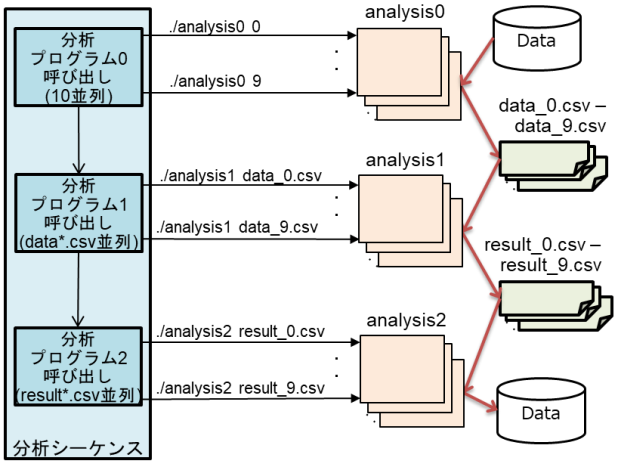

図 3 分析シーケンス実行イメージ

が見込まれる。これに対し、各サーバにあらかじめ分析プログラムを登録しておき、必要な並列実行数に応じて分析プログラムの実行を指示することで、低い処理負荷で処理を並列実行することが可能であると考えられる。

以上の着眼点に基づいた、提案する分析プラットフォームの全体像を図 2 に示す。本分析プラットフォームは、分析プログラム、分析シーケンス、分析プログラム実行基盤、分析シーケンス実行基盤、管理プログラムの 5 つの要素で構成する。分析プログラムはデータ分析やデータクレンジング、データ可視化などに用いる分析ツールを用いたプログラムそのものを表す。分析シーケンスは、分析プログラムの実行順序及び各分析プログラム並列実行数を記載したシーケンスである。分析プログラム実行基盤は、すべてのサーバにあらかじめ搭載するプログラムであり、分析プログラムの起動を担う。分析シーケンス実行基盤は、分析シーケンスに基づいた分析プログラムの実行を制御する。最後に、管理プログラムは、分析シーケンス実行基盤及び分析プログラム実行基盤を管理するための仕組みである。管理プログラムについては次節で詳細の説明を行う。

3.2 全体動作概要

分析シーケンス実行基盤は、分析シーケンスに基づき、分析プログラム実行基盤に対して実行する分析プログラム名およびパラメータを渡し、分析プログラムの実行を指示する。分析プログラム実行基盤は、受け取ったプログラム名とパラメータに基づいてシェル経由で分析プログラムを起動する。分析プログラム実行基盤の動作は、プログラム名およびパラメータを受け取り、分析プログラムを起動を実施するのみであるため、処理負荷が低い。ここで、分析シーケンス実行基盤は、同一の分析プログラムをパラメータを変えつつ複数同時に分析プログラム実行基盤に実行を指示することが可能である。そのため、分析プログラムのパラメータに例えば処理対象のデータを含めることで、同一の分析プログラムを用いた異なるデータに対する処理を並列に実行することが可能となる。たとえば、分析プログラムを 10 並列実行する場合に、各分析プログラムに 0 から 9 までのパラメータをそれぞれ渡すことや、特定のディレクトリのファイル数分並列で実行する場合に、当該ディレクトリ内のファイル名をそれぞれパラメータとして渡す事が可能である。

分析シーケンス実行基盤は、分析プログラムの実行指示ごとに、指示先の分析プログラム実行基盤を選択する必要がある。選択方法としては例えば、ラウンドロビンに選択する方法や、分析プログラム実行基盤の動作するサーバの CPU 負荷に応じて選択することが考えられる。本プラットフォームでは、前述のような一般的な選択方法はデフォルトとして提供する共に、別途選択方法をプログラムを実装することで、必要に応じて分析プログラム実行基盤の選択方式を別の方式に置き換えることも可能とした。これにより、例えば処理ごとに優先的に割り振りたいサーバを定めることなどが可能である。なお、分析プログラム実行基盤は最大で実行可能なプログラム数の設定を保持しており、分析シーケンス実行基盤は、実行可能なプログラム数を超えない範囲で分析プログラム実行の指示を行う形とした。

以上をまとめると、分析プラットフォーム全体の動作の流れは次のようになる。まず、分析シーケンス実行基盤は、分析シーケンスを読みだし、実行する分析プログラム名及び並列実行数を判別する。そして、各サーバの分析プログラム実行基盤に対し分析プログラムの実行を指示する。各分析プログラム実行基盤は実行指示を受け取ると、分析プログラム名とパラメータを元に、シェル経由で分析プログラムを実行する。分析プログラムの実行が終了すると、分析シーケンス実行基盤に、プログラムの実行が終了した旨を通知する。分析シーケンス実行基盤は分析シーケンス実行基盤においてすべての分析プログラム実行指示の完了を確認した後、次の分析プログラムの実行の指示を分析プロ

グラム実行基盤に対して実施する。

分析シーケンスの実行例を図 3 示す。本例ではまず、分析シーケンス実行基盤は、分析プログラム 0 を 10 並列で、各プログラムに渡すパラメータを 0 から 9 として分析プログラム実行基盤に対して実行を指示する。各分析プログラム 0 は、data_*.csv を作成する。次に、分析プログラム 0 の実行が終了した後に、分析シーケンス実行基盤は、分析プログラム 1 を、分析プログラム 0 が作成した data_*.csv のファイル数分並列に、本ファイル名をパラメータとして並列に、分析プログラム実行基盤に対して実行を指示する。最後に、分析シーケンス実行基盤は、分析プログラム 2 を、分析プログラム 1 が作成した result*.csv 数分並列に、本ファイル名をパラメータとして分析プログラム実行基盤に対して実行を指示する。図 3 に示す例では、分析プログラム間の並列度を同一とし、並列実行単位ごとにデータを引き継いでゆくように設計している。

以上の仕組みに基づくことで、分析プログラム実行基盤を導入した各サーバ上で任意のシェルから実行可能な分析ツールを並列に実行することが可能であり、また、分析シーケンスを書き直すことで、実行する分析プログラムや並列実行数を簡単に変更することが可能となる。また、分析プログラム実行基盤は、分析シーケンス実行処理基盤の指示を受け、シェルを起動するのみであるため、非常に処理を軽くすることが可能である。さらに、必要に応じて実行する分析処理基盤を選択するプログラムを実装することで、プログラム実行サーバも柔軟に選択することが可能である。

3.3 クラスタ構成管理の導入

本分析プラットフォームでは、分析シーケンス実行基盤が分析プログラム実行基盤を動作するサーバ情報を入手し、分析プログラムの実行の指示を行う必要がある。分析プログラム実行基盤が動作するサーバは実行途中に故障する可能性や、保守のために一部のサーバを停止することがあるため、分析シーケンス実行基盤は各分析プログラム実行基盤のサーバの状況を取得し、動的に分析プログラム実行の指示先を判断する必要がある。

上記を実現するために、zookeeper[7][8]を用いた分析管理プログラムを導入する。まず、分析プログラム実行基盤は起動時に zookeeper に自身の動作するサーバの IP アドレスなどの情報を登録すると共に、zookeeper と keepalive のセッションを張り、例えば 2 秒ごとに相互に監視を行う。分析プログラム実行基盤停止時は、zookeeper より自身のサーバの情報を削除する。分析プログラム実行基盤やサーバが故障した場合は、zookeeper は keepalive のセッション切断を検出し、zookeeper が分析プログラム実行基盤の情報を削除する。以上の動作により、zookeeper 経由で、分析シーケンス実行基盤は分析プログラム実行基盤が動作している全てのサーバの情報を得ることが可能である。また、zookeeper はサーバが断続的に、中途半端に故障した場合や、

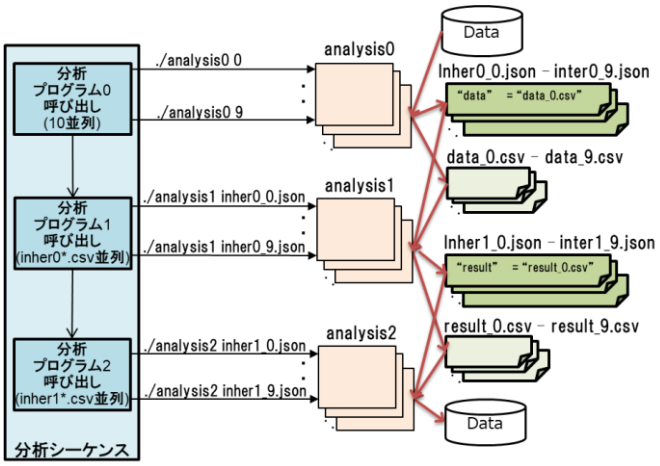

図 4 引き継ぎファイルによる I/F 方式標準化

ネットワーク障害によって sprit brain が発生した場合においても、正しくサーバ障害を検出することが可能であり、優れた耐故障性を保証することが可能となる。

3.4 I/F 方式の標準化

本プラットフォームを用いた場合、分析プログラムは起動時に渡されるパラメータごとに、例えば処理対象のデータを変化させる必要がある。そのため、分析シーケンスは分析プログラムに対する I/F、すなわちパラメータを意識した上で実装する必要がある。しかし、分析シーケンスや分析プログラムは必ずしも同一の開発者が実装するとは限らず、開発者間の I/F の意識合わせが必要となる。通常のシステム開発では I/F を一度決定した後、I/F を変更することは少ないので、I/F の決定に時間を掛けても問題は無かった。しかし、分析システムでは何度も分析システムを変更することで、分析施行による仮説検証を繰り返すという特徴上、I/F も何度も変更する必要が生じ、課題となる。

これに対し、分析プログラムに対するパラメータをすべて記載した引き継ぎファイルを新たに用意し、分析シーケンスから分析プログラムに対して引き継ぎファイル名を渡すことで、I/F を標準化する方式が考えられる。図 3 に示す構成を、本仕組みを用いて実装した場合の例を図 4 に示す。本仕組みを用いた場合、分析シーケンスと分析プログラム間の I/F は引き継ぎファイル名として固定化する。そして、分析プログラムはプログラムの冒頭で引き継ぎファイルを読み出し、自身に対するパラメータとして利用する。また、分析プログラム実行終了時に、次に実行する分析プログラム向けに、新たな引き継ぎファイルの生成を行う。例えば、図 4 の例では、analysis1 は analysis0 の出力である data*.csv を利用する。そのために、analysis0 は inher0*.json に data*.csv という情報を記載し、引き継ぎファイルとして出力する。analysis1 は inher*.json を読み出し、自身に対する入力となる data*.csv というファイル名を知ることができる。このようにすることで、分析シーケンスにおける I/F は固定することが可能となる。例えば、図 4 の例において、新たに analysis0 と analysis1 の間で別のファイルのやり取

りが必要になったとしても、分析シーケンスは引き継ぎファイル inher0*.json をパラメータとしたままでよく、analysis0 が inher0*.json に新たに analysis1 に渡すデータのファイル名として例えば datanew*.csv を含めればよい。

また、分析シーケンスより起動する最初のプログラムが後段で必要な並列実行単位ごとのパラメータを記した引き継ぎファイルを特定のディレクトリに準備し、その後は当該ディレクトリに含まれるファイル数分だけ並列実行を繰り返すように実装すれば、分析シーケンスの実装形態は非常に単純になり、分析施行のたびに分析シーケンスにおける I/F の検討や分析シーケンスの設計を行う必要がなくなる。但し、分析プログラムによる引き継ぎファイルの読み出し及び作成を行う必要が生じる点がデメリットとなる。

4. 評価

4.1 評価環境

表 1 各サーバのスペック

CPU	2 core processor (3.06GHz)
Memory	DDR3 8GB x2
OS	Linux® 3.10.0
HDD	20GB

複数台のサーバを用意し、各サーバ上に分析プログラム実行基盤及び分析プログラムを導入した環境で評価を行った。各サーバの 1 台あたりの環境を表 1 に示す。サーバとして、分析プログラム実行基盤及び分析プログラムを導入した分析サーバを 5 台、管理サーバを 1 台用意した。分析シーケンス及び分析シーケンス実行基盤も 5 台のサーバに導入し、いずれのサーバからであっても分析シーケンスを実行可能な形とした。

4.2 評価システム

本分析プラットフォーム評価の為に、シミュレータを用いてデータを生成し、当該データに対して仮想的な分析を行なった。

データは、工場内のモノ(Object)及び場所(Place)に ID を振り、時間ごとにどの ID のモノがどの ID の場所にいるかを示すものを想定した。図 5 の左側に入力データの例を、右側に出力データの例を示す。例えば、入力データは、2018/04/01 10:00:00 に、ID#5 の物は、ID#8 の場所に存在することを示し、また、出力データは、例えば場所の ID#3 には、各モノは平均で 30 分滞留していることを示している。今回は、一か月分のデータとして約 3GB の入力データを作成した。入力データは、Elasticsearch®[9]に格納し、評価を行った。本入力データを用い、下記のような流れで分析を行い、出力データを得るシステムを構築した。まず、Elasticsearch®から必要なデータを抽出し、ファイル形式として格納する。次に、取り出したデータに対して統計処理

@timestamp	Object	Place
2018/04/01 10:00:00	5	4
2018/04/01 10:00:00	6	3
	・・・	
2018/04/01 10:00:01	5	5
2018/04/01 10:00:01	6	3
	・・・	

Time-range	Place	Average process time (min)
2018/04/01 10:00:00 - 2018/04/01 11:00:00		・・・
	3	30
	4	5
	5	10
		・・・
	・・・	

図 5 分析対象入力)データ及び出力データ

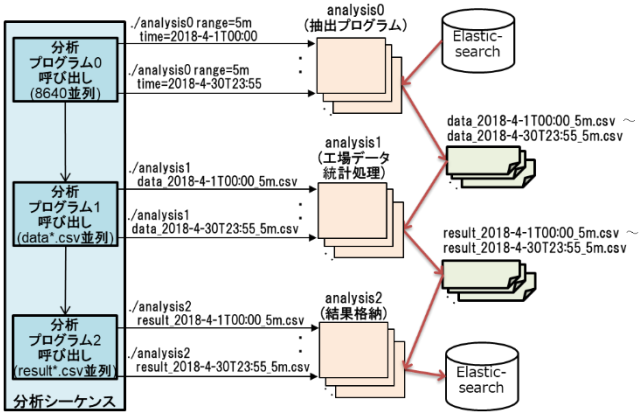

図 6 評価システム構成

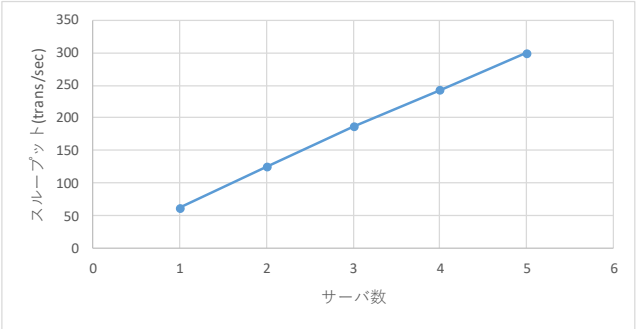

図 7 サーバ台数に応じたスループット

し、特定の時間範囲ごとに各 ID のモノがどの程度の時間、ある ID の場所に残留しているかを算出し、場所ごとの平均を取得する。最後に、分析結果を Elasticsearch へ格納する。分析プログラムは、データ抽出、データ統計処理、データ格納の 3 つのプログラムで構成し、モノの ID、場所の ID、時間範囲データで区切り、区切ったデータごとにプログラムを並列に実行した。

実際に組んだ評価システムの構成を図 6 に示す。まず、データ抽出プログラムに対し、時間範囲を 5 分間ごととして、抽出プログラムを 30*24*60/5=8640 並列で実行する。抽出プログラムは対象のデータを含むファイルを特定のディレクトリに例えば data*.csv として生成する。次に、data*.csv ファイル名をパラメータとして、統計処理プログラムを並列に実行する。統計処理プログラムは、結果を別のディレクトに例えば result*.csv として生成する。最後に、result*.csv ファイル名をパラメータとして、データ格納プログラムを並列に実行する。

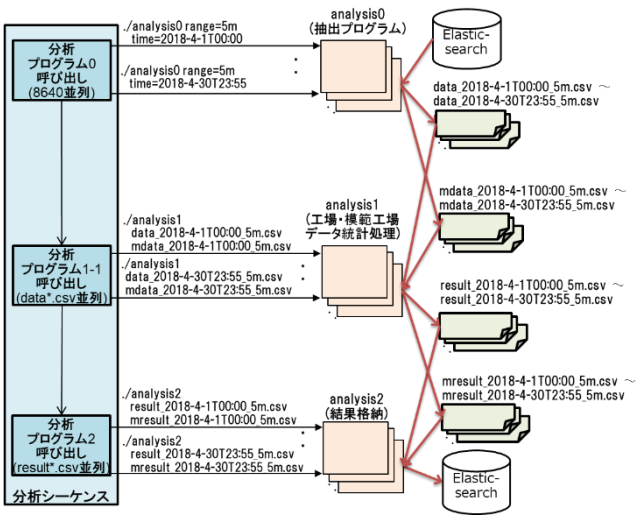

図 8 変更後のシステム構成

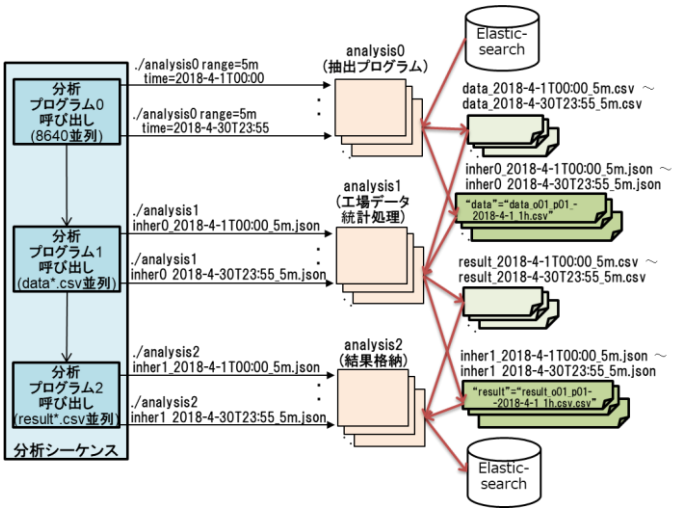

図 9 評価システム構成(I/F 標準化手法利用時)

4.3 高速性の検証

まず、分析に利用するサーバ台数を 1 台から 5 台それぞれで分析を実行し、それぞれの台数のサーバを用いた場合のスループットを測定した。測定結果を図 7 に示す。

本グラフより、例えばサーバ 1 台の場合のスループット 62trans/sec に対し、サーバ 5 台の場合にスループットがおよそ 5 倍の 300trans/sec であり、サーバ台数に応じて実行性能がスケールしている事が分かる。すなわち本分析プラットフォームによる並列化実行機能によって、サーバ台数を増やせばプログラムを高速に実行することが可能となり、並列実行による高速化を実現出来たと考えられる。

また、分析プログラム 0、分析プログラム 1、分析プログラム 2 をそれぞれ 1 並列で 1core で順番に、本システムを利用せずに順々に実行した場合のスループットはおおよそ 31trans/sec であった。サーバ 1 台 2core の場合のスループット 62trans/sec と比較すると、本プラットフォームを利用することによる負荷はほぼ無いということがわかる。

4.4 柔軟性の検証

4.2 で述べた評価システムに対し、仮説検証ごとのシステムの組み換えを想定し、その組み換えの際に必要な作業時間を算出した。変更としては、模範となる工場を想定し、2 つの工場に対して同一の統計処理を行い、工場が模範工場に対してどの程度ずれているのかを算出し、結果を格納するように変更を行った。そして、変更に必要となる工数を算出した。この際、I/F 標準化手法を用いて設計した場合と、用いずに設計した場合それぞれに対し、実際に実装を行い、工数の算出を行った。

I/F 標準化手法を用いない場合の変更後のシステムの構成図を図 8 に示す。変更後のシステムでは、最初に呼ばれる抽出プログラムで工場のデータに加えて模範工場のデータ(mdata_**.csv)を抽出していること、工場・模範工場データ統計処理プログラムで、工場のデータに加えて模範工場のデータも読み込み、模範工場の結果(mresult_**.csv)も出

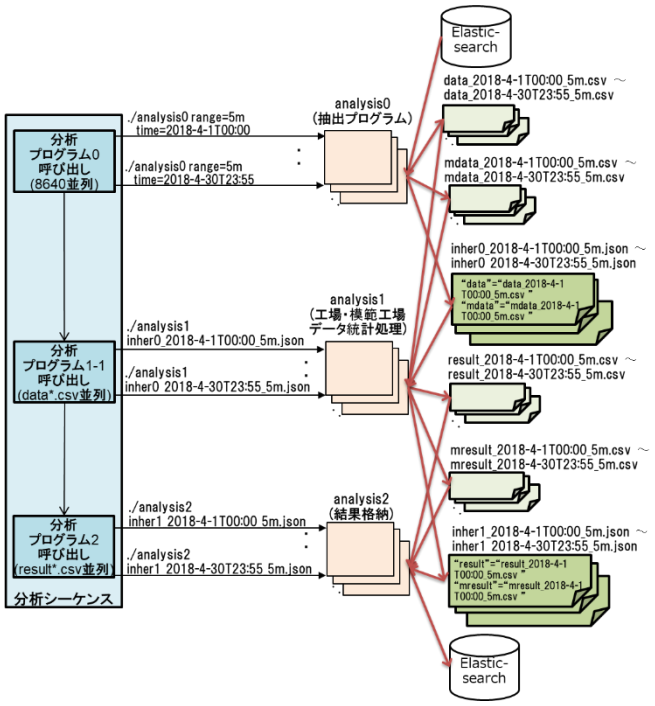

図 10 変更後のシステム構成(I/F 標準化手法利用

力していること、結果格納で工場の結果に加えて模範工場の結果も出力していこと、が異なる。そのため、図 7 と図 8 を比較するとわかるように、分析シーケンスから各分析プログラムを呼び出す際の I/F に、模範工場のデータ及び模範工場の結果を含める必要が生じ、それに合わせて I/F の変更を行っていることがわかる。

一方、I/F 標準化手法を用いた場合の、変更前のシステム構成図を図 9 に、変更後のシステム構成図を図 10 に示す。I/F 標準化手法を用いた場合であっても、抽出プログラム、工場・模範工場データ統計処理プログラム、結果格納プログラムそれぞれで、模範工場のデータもしくは模範工場の結果を扱う必要が生じるのは、I/F 標準化手法を用いなかった場合と同様である。しかし、I/F 標準化手法を用いた場合

は、抽出プログラムと工場・模範工場データ統計処理プログラム間の引き継ぎファイル(inher0_**.json)に模範工場のデータのファイル名(mdata_*.csv)を、工場・模範工場データ統計処理プログラムと結果格納プログラム間の引き継ぎファイル(inher1_**.json)に模範工場の結果のファイル名(mresult_*.csv)を加えるのみでよく、分析シーケンスと分析プログラム間の I/F は、変わっておらず、修正不要であることがわかる。

I/F 標準化手法を用いた場合と用いなかった場合それぞれのにおいて変更に必要な工数の延べ人数を、実際に実装を行って算出した結果を、表 2 に示す。本プラットフォームを活用した場合、I/F 標準化手法を用いずとも 4 日でシステムの変更が可能であり、変更工数は十分に工数が小さいといえる。I/F 標準化手法を用いた場合は、作業時間を 4 日から 3 日へ、約 30%削減できることが分かった。表 2 からもわかるように、主な削減ポイントは、I/F 部分の検討及び実装工数であり、I/F 標準化手法を用いることで、分析シーケンスと分析プログラムの I/F 部分の実装が不要になったため、大きく工数を削減することができたと言える。

表 2 変更工数

	プラットフォーム活用	I/F 標準化仕様利用
シーケンス検討	0.5day	0.5day
I/F 検討及び実装	1.5day	0.5day
プログラム実装	2day	2day
合計	4day	3day

5. 結論

大量にデータを分析する分析システムにおいて、様々な分析ツールを簡単に組み合わせることを可能にする柔軟性と大量のデータを高速に分析可能な高速性が求められている。本報では分析をシーケンスと管理することで分析プログラムと並列実行数の変更を容易化し、分析プログラム実行基盤を各サーバに導入した上で各分析プログラム実行基盤に分析プログラム実行の指示を並列に実施することで、分析ツールの複数サーバ上での並列処理を低負荷で実現する、分析プラットフォームを提案した。シミュレータによって生成したデータを用いた分析システムを本プラットフォームを用いて構築し、評価を行った結果、サーバ数に応じてスループットがスケールすること及び本プラットフォームによる負荷が小さいこと、また、分析システムの実装変更時の工数が 4 日と十分小さいことを確認した。さらに、I/F 標準化手法を用いることで、実装変更時の工数を 3 日とさらに削減すること可能であることを確認した。

今後の課題としては、分析プログラム実行基盤の導入方式の容易化や、分析シーケンステンプレート化の検討など、多岐に渡ると考えている。

参考文献

[1] Tomohiro Fukuhara, Ryuhei Tenmoku,Ryoko Ueoka, Takashi Okuma and Takeshi Kurata, "Estimating skills of waiting staff of a restaurant based on behavior sensing and POS data analysis: A case study in a Japanese cuisine restaurant," Proceedings of the 5th International Conference on Applied Human Factors and Ergonomics, pp.4287-4299, 2014.

[2] I.O. Pappas, P.E. Kourouthanassis, M.N. Giannakos, V. Chrissikop oulos, “Explaining online shopping behavior with fsQCA: The role of cognitive and affective perceptions,” Journal of Business Research, Vol.69 Issue2, pp.794-803, February 2016.

[3] Jiaming Fang , Benjamin George , Yunfei Shao , Chao Wen, “Affective and cognitive factors influencing repeat buying in e-commerce,” Journal of Electronic Commerce Research and Applications, vol.19 Issue C, p.44-55, September 2016.

[4] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica., “Spark: cluster computing with working sets.,” In Proceedings of HotCloud ’10, 2010.

[5] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica., “Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing.,” In Proceedings of NSDI, pages 15–28, 2012.

[6] Mauricio A. Hernández , Salvatore J. Stolfo,, “Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem,” Data Mining and Knowledge Discovery, vol.2 number.1, pp.9-37, January 1998

[7] Hunt, P., Konar, M., Junqueira, F. P., and Reed, B., “ZooKeeper: wait-free coordination for internet-scale systems.,” In Proceedings of USENIX Annual Technical Conference, pp. 145–158, 2010.

[8] Junqueira, F. P., Reed, B. C., and Serafini, M., “Zab: High-performance broadcast for primary-backup systems.,” International Conferenace on Dependable Systems & Networks, IEEE Computer Society, pp. 245–256, 2011.

[9] Open Source Search & Analytics · Elasticsearch | Elastic, https://www.elastic.co/