IoT システムにおけるビッグデータの可視化集約と操作連動

Visualization Integration and Operation Linkage for Big Data from Large IoT System

李新肖　黒田亮

Xinxiao Li,　Akira Kuroda

株式会社東芝　研究開発センター
Research and Development Center, Corporation Toshiba

Abstract: In a large scale IoT system, structured and unstructured data are collected from various distributed sensors. It is important to visualize these data with an overview context composed of multiple views and interactively focus on some detail views to understand the current system status. But a unified VA (Visual Analysis) is difficult owing to due to being short of expressly relationship between distributed datasets from different sensors or processed subsets of big data. In this paper, we present a visualization framework to analytically acquire the relationship among distributed or processed subsets, integrate their views in a visualization context, and realize operation linkage between them.

前言

大規模 IoT システムでは、分散された各種のセンサーにより収集されたデータとその分析処理結果を可視化して、システム全体の状況を把握することが重要である。センサーの分散化と通信スピードの向上に伴い、IoT システムでは、時系列データ、音声データ、画像データなどの構造データと非構造が混在するシステムは多い。5V（Volume・Velocity・Variety・Veracity・Value）が特徴とするビッグデータの入手が容易になっている一方、トライ・アンド・エラーを繰り返してデータを視覚的に分析（VA:　Visual Analysis)することが困難である。この主な原因は、データ量増大によるデータ転送や描画のコストが増大したことと、多種多様なデータを扱うことによる関連性の乏しいデータ間の可視化連動ができないことである。

本論文では、VA を実現する IoT システムの可視化フレームワークを提案する。

BI と AI の融合：必要性

可視化中心の BI（Business Intelligence）と機器学習中心の AI（Artificial Intelligence）が分断化することなく、シームレスに融合することは、IoT システムにおけるデータ分析の極意であり、そのような VA システムが望ましい。従来の IoT システムは、可視化と分析のどちらに偏り、直感的なシステム俯瞰と深い分析が両立できない。

例えば、製造や産業の現場でプロセス制御と集中監視を行う監視制御システム、収集したセンサーの情報を可視化することを中心としている。全体の状況をリアルタイムに把握し、人間の目で状況を判断して処置することが基本となる。

また、セルフ BI システムは、データソースとチャートの多様性と操作性を重視し、分析機能が不足している。

一方、多くの機械学習システムは、センサーデータを分析して、その結果を静的なものとして提示する。例えば、カメラセンサーや音声センサーから収集したデータを、CNN(Convolutional Neural Network)や LSTM（Long short-term memory）などの機械学習により分析結果を BI により可視化する。これらの方法は、分析と可視化が分断されており、より深いデータ分析を妨げている。

また、ドメイン知識とカスタマイズされた機械学習モジュールの組み合わせた全自動分析方法で得られた分析結果は、生データを見ることより深い知識が得られやすくなっているが、システムの一部データしか対応できず、全体が把握しにくい欠点がある。

現実課題と本論文の提案方法

センサーからデータを収集して全体の状況把握に使われる IoT システムでは、センサー数の増大や非構造データ（音声やビデオ）によりデータ転送遅延の問題が顕在化する。

現実の複雑なビッグデータの分析は、ドメインエ

キスパート（エンドユーザになる場合が多い）、データエンジニア、ソフトウェアエンジニア、データアナリストなどの分業と協同により実現される。エンドユーザに、ML モデル構築などデータ分析スキルを要求することが無理である。しかし、データ分析結果を即時に BI の可視化対象データとして利用することが AI と BI のエンドユーザにとって重要である。

本論文で提案する IoT システムの可視化フレームワークは、分散されたセンサーに近い場所（エッジとは限らない場所）に ETL（Extract・Transform・Load）処理モジュールや ML 処理モジュールを配置し、分散された処理モデルの分析結果を集めて可視化することで、データ可視化とデータ分析を統一的なフレームワークで扱う。データの処理結果を転送するため、元のデータを転送することよりデータ転送遅延の問題が緩和される。

多くの商用セルフ BI システムは、データの可視化結果を表す各ビューの操作連動を実現するため、各ビューに用いるデータセット間で共通キーが必要である。その共通キーは、データセット間に共通カラムであるか、ユーザがカラム間の対応関係を指定することで実現する。しかしながら、複雑なデータの関連性は、機械学習に介在しないと実現できないことがある。また、同一の IoT システムでも、可視化するために転送される複数の部分データセット同士は、直接関連性が無く、元のデータセットにたどり着かないと、関連データが得られない。

本論文の提案する可視化フレームワークは、処理モデル間の連携に介在し、可視化端末の複数ビューに対応するデータセット間の潜在関連性を見つけ、ビューの操作連動を実現する。

システム構成

図 1 はシステム構成図であり、システムは「可視化集約＆操作連動層」、「分散＆連携処理層」、「データソース層」で構成される。

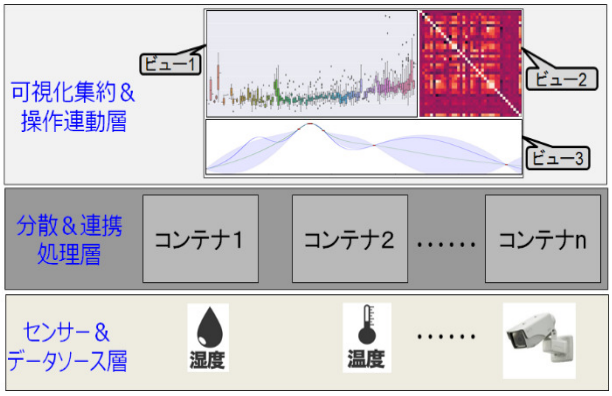

図 1　システム構成

「データソース層」は、分散されたデータ供給源を示す。センサーとデータベースの組み合わせにより現在と過去に収集されたデータを供給する。分散された各種センサーは、同じ IoT システムに属するか、同じタイプのデータかで、何らかの関係があると仮定する。例えば、ビル監視システムの属するセンサー群、心電図などを取る装置などである。

「分散＆連携処理層」は、通信遅延の抑制のため通信量を最小化する処理を実現する必要がある。データソース層からを必要な部分データを抽出し、必要な形に加工して、「可視化集約&操作連動層」に供給する役割を示す。「データソース層」にあるビッグデータを入力とし、可視化に必要な最小限のデータに変換して「可視化集約＆操作連動層」にデータを供給することが望ましい。

また、少なくとも、中間層（「分散＆連携処理層」）は、データ通信のフラット化により通信遅延を低減し、可視化画面の操作のレスポンスを 1 ～ 3 秒以下に抑えることを目標とする。

中間層は分散された異なるデータ処理モジュール（本文ではコンテナと呼ぶ）に構成され、それぞれの提供する可視化機能に応じて、独自のデータ処理ロジックやデータ加工ロジックを持っていて、データの ETL 処理、分析処理や可視化端末側とやり取りの機能を実現する。それらのデータ処理モジュールは可視化に必要なデータを抽出して、適切な可視化フォーマットに加工して、可視化側に供給する。

「可視化集約＆操作連動層」は、可視化画面生成リクエストに応じて、下層から可視化対象のデータセットをもらい、各ビューを生成し、さらに、ダッシュボードを生成する（本文の以下では、画面とダッシュボードが同義として混用する）。この層は、可視化対象データをまとめて描画する可視化集約機能を実現する。

この層は複数の可視化端末を分散して、どこでもアクセスしやすいため、Web ブラウザで実現することが望ましい。可視化対象データをまとめて Web ブラウザへ転送する機能は、Web サーバの一部機能とし、通常の Web サーバと合併する。さらに、システムが言語や環境非依存の疎結合を実現するために、センサー＆データソース層と中間層の間に、MQ（Message Queue）を介在して良い。ここでは、MQ より可視化対象データ転送の細かい実施方法の説明を省略する。

操作連動方式

図 2 は二つのコンテナの連携によりビューの操作連動方法を示す。

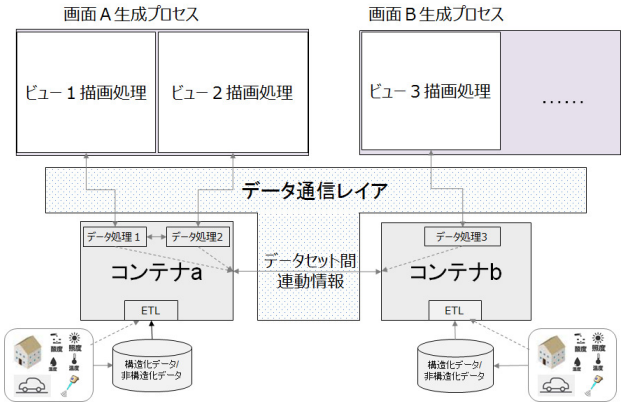

図２　ビュー間の操作連動

画面 A 生成プロセスと画面 B 生成プロセスはそれぞれ、ビュー１～２とビュー３の描画処理を持つ。コンテナ a、コンテナ b はそれぞれデータ通信レイアを介して、ビュー１～３に可視化対象データセットを供給する。ビュー１～３の操作情報（データの絞り込み範囲などのパラメータ）は、コンテナ a～b に送信する。コンテナ a とコンテナ b は互いに、データのスキーマ情報、操作情報、データ分析結果を参照できる。二つのコンテナは、一方の可視化対象データの変動が他方の可視化対象データセットを連動させ、さらに、各ビューの描画処理に通して、ビューを連動させる。ビュー操作連動案は図３（案１）と図４（案２）に示す。

図３の情報やり取りステップ：

① 可視化対象データセットをコンテナから画面生成プロセスに送信し、画面（A、B）を生成する。
② ビュー１のユーザ操作情報をビュー１から対応しているコンテナ a へ通知する。
③ コンテナ a からコンテナ b へ操作情報を含んだコンテナ連携情報を送信する。
④ コンテナ a、b のデータ処理プロセスにより双方の可視化対象データセットを更新する。
⑤ 各コンテナに対応するビュー（ビュー１～３）を更新する。

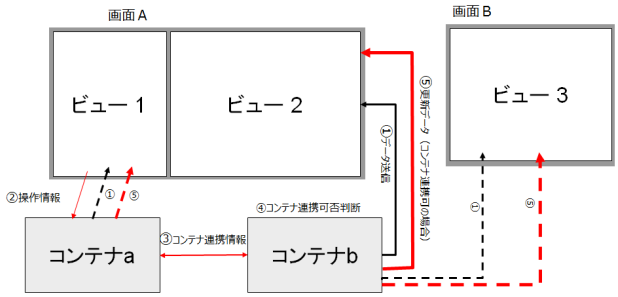

図３　操作連動の情報やり取り（案１）

案１はユーザのビュー操作イベントにトリガーされ、コンテナ a、b の間に連携情報を伝播し、コンテナ間の連携が行われる。

一般的には、コンテナ側に計算リソースが集中するため、頻繁な情報のやり取りが不利である。下記の案２は、各コンテナが事前にほかのコンテナとの連携可否情報を収集して、連携できるコンテナリストをビュー側に送信する。ユーザのビュー操作イベントにトリガーされ、操作されたビューは連携情報を伝播する。その連携情報を受信した各ビューはそれぞれの対応しているコンテナ側にその連携情報を送信し、コンテナ間の連携が行われる。さらに、コンテナ間の連携に介して、可視化対象データ、または、各ビューを更新する。ビューに構成されたユーザインターフェース側からみれば、一方のビュー操作は、他方のビューの更新と連動している。

図４の情報やり取りステップ：

① 連携コンテナリスト作成：　各コンテナが定期的にデータスキーマを含んだ情報をほかのコンテナに伝播する。各コンテナが、どのコンテナと連携できるか、を各自の組み込んだロジックで連携コンテナリストを作成して保持する。
② 可視化画面を生成する際に、連携コンテナリストとデータセットを各コンテナから各画面生成プロセスに送信し、各画面（画面 A、B）を生成する。
③ ビュー１の操作情報を連携コンテナの対応しているビュー（ビュー２、３）に送信する。
④ ビュー１～３は、操作情報を各自の対応しているコンテナに送信する。
⑤ 各コンテナが操作情報により可視化対象データを更新し、これらの更新データをビューに返信し、ビューを更新する。

ビューとコンテナの組み合わせは 1 対 N と N 対 1 にすることができる。

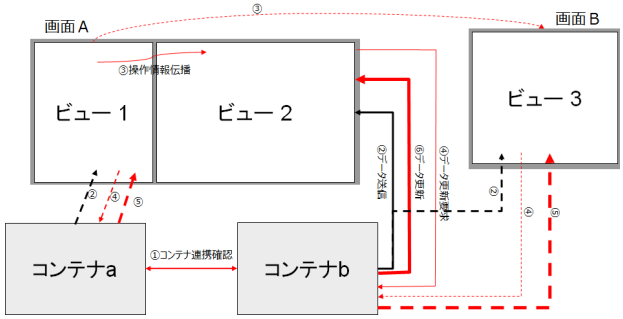

図４　操作連動の情報やり取り（案２）

コンテナ連携可否の決め方

各コンテナが独立な存在で、１つのコンテナの起動、停止やコンテナの内部の処理ロジックの更新が

他のコンテナに非依存である。コンテナ連携の可否は、他のコンテナ連携情報を受け取り、コンテナの内部ロジックで判定する。この判定処理は、ML ベースやルールベース（エキスパートシステム、ドメイン知識データベース）でのどちらも可能である。図５は、ML ベースのコンテナ連携例である。

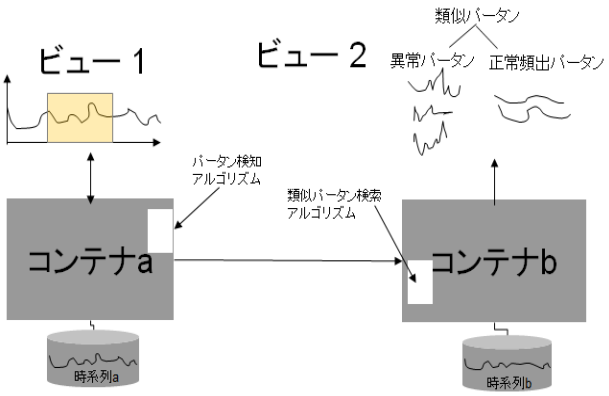

図５　コンテナ連携の例

図５の例では、コンテナ a、コンテナ b がそれぞれ心電図である時系列１、時系列２のデータを可視化端末に送信する。可視化端末の画面描画プロセスは、時系列１と時系列２を受信し、折れ線グラフ（ビュー１）と、類似パータンツリー（ビュー２）を生成する。時系列１と時系列２は心電図データであることを共通し、それ以外の明確な共通情報がないとする。ビュー１の画面操作により、時系列１のフォーカス範囲がきめられ、そのフォーカス範囲をコンテナ a に送信する。コンテナ a は、フォーカス範囲を受信し、パータン検知アルゴリズムにより、フォーカス範囲の時系列１のデータから異常パータンと頻出パータンを検出する。さらに、コンテナ a は、検出した異常パータンと頻出パータンをコンテナ b に通知する。コンテナ b は、コンテナ a から通知イベントにトリガーされ、そのコンテナ a のパータンの詳細を参照し、時系列２のデータから、類似する異常パータンや頻出パータンを検出する。コンテナ b は、さらに、検出した異常パータンや頻出パータンを、コンテナ b の対応している画面描画プロセスに送信する。画面描画プロセスは、パータン検出のデータを受信し、ビュー２に更新する。可視化端末から見ると、ビュー１の操作を操作することにより、ビュー２が連動して更新される。

上記の例では、ユーザインターフェース側からみれば、ビュー１とビュー２は明確な共通情報がないが、それぞれの対応しているコンテナ側での心電図時系列のパータン検索アルゴリズムの連携により、ビュー１の操作によりビュー２の連動が実現できる。各コンテナは、メモリと計算パワーを高いマシンで、ドメイン知識、ルール、複雑な機械学習アルゴリズムを組み込むことができる。

画面内連動をコントロールする

各ビューは連携コンテナリストにより自動的に連動することが可能となるが、、ビューの勝手な連動はエンドユーザにとって不都合である場合が多い。連携コンテナリストをベースに、ビューの間に連動可否をエンドユーザにコントロールされることは、セルフ BI をはじめ、視覚的データ分析において重要である。

画面内各ビューの連動は、有向グラフで表すことができる。例えば、図６のビュー連携有向グラフの例である。矢印の方向は、操作情報の伝播方向を示す。ビューが自動的に連動するなら、図中のように、ビュー１→ビュー２→ビュー１と、ビュー１→ビュー３→ビュー４→ビュー１が無限ループになる問題がある。

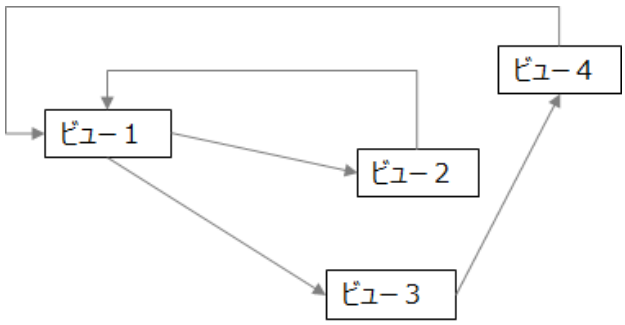

図６　画面内ビュー間連動有効グラフの例（矢印はビュー間の操作イベント伝播方向を示す）

連携コンテナリストに対応するすべてのビューは連動候補ビューリストとする。上述の無限ループを断ち切るために、下記のビュー連動条件を設ける。

（１）連動候補ビューリストの中からユーザに連動を許可するビューのみを連動させる。この条件は図７で詳しく説明する。

（２）連動をさせたビューへ連動情報の伝播は、１階層のみに限定する。例えば、図６のビュー３→ビュー４→ビュー１の連動有向パスがあっても、ビュー３の操作イベントは、ビュー４へ伝播し、ビュー１に伝播することはしない。

上記二つの条件により、ユーザがビュー間の連動を完全にコントロールすることができる。例えば、図６のような任意なビュー連動有向グラフを構築するために、図７の対話型 UI（User Interface）で設定する。

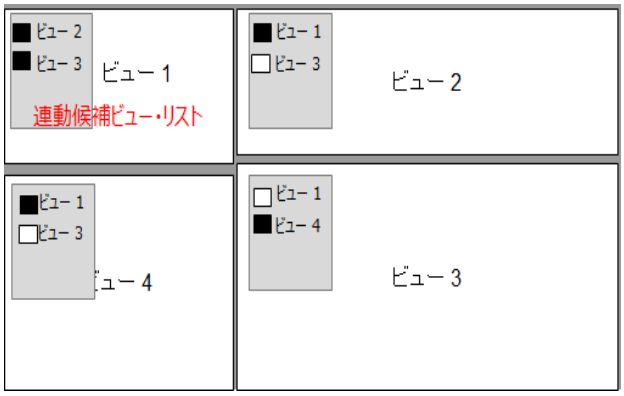

図７　画面内ビュー連動設定 UI の例

図７は、図６の連動有向グラフを作成するためのビュー連動設定 UI である。各ビューに、マウスクリックなどの操作でビュー連動候補リストのポップアップメニューを出して、チェックボックスにより連動可・不可を簡単に手軽に操作できる。例えば、ビュー１と連動できる候補ビューは、ビュー２とビュー３であり、双方ともチェックされ、連動を許可している；ビュー３の連動候補ビューは、ビュー１とビュー４であり、ビュー４のみの連動を許可している。

画面間連動をコントロールする

データ視覚的分析のシナリオによって、ユーザインターフェースは複数の画面に構成されるケースがある。ビュー操作により画面に跨ってビューを連動させる必要である。画面間のビュー間連動は、画面内のビュー間連動と同じように、画面間の連動順（画面間の操作情報連動順）は、データ分析のシナリオに従い、ユーザにコントロールされることが望ましい。

画面間の連動順は画面間連動有向グラフにより表現される。画面間連動順はユーザが画面連動の有向グラフを作成することによりコントロールする。

図８は、画面連動順の基本パータンを示す。画面間の連動順は三つの基本パータンに分ける、それぞれ、順番連動、分岐連動とループ連動である。すべての画面連動有向グラフはその基本パータンの組み合わせで表すことができる。

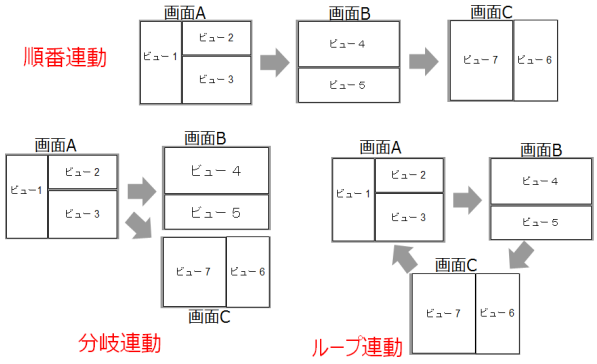

図８　画面連動順の基本パータン

画面内ビュー間連動と同じように、複雑な画面間連動順が無限ループを形成される可能性がある。その無限ループを断ち切るために、画面内のビュー間連動のように、画面間の連動は作成された有向グラフ順に１階層しか伝播しない、また、許可した画面ペア間しか連動させない制限条件を設ける。つまり、図９の例で示すように、画面 A の操作を画面 C と連動させ、画面 B の操作で画面 C と連動させる設定で、その連動情報の伝播は設定した画面間と方向のみに許可する。自動的に画面 A の操作で画面 C へ連動させない。また、画面連動有向グラフの矢印の逆順に連動させない。

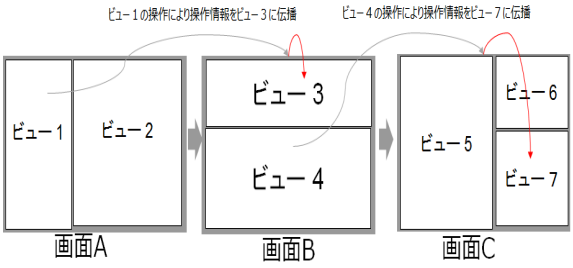

図９　画面間連動の例

ユーザ操作によって画面間連動は、画面内のビュー連動によりやや複雑になる。ユーザの操作単位がビューであるため、操作連動が基本的にビューの間の連動である。画面間の操作連動は、まず、画面に跨って連動情報を伝播し、次に、その連動情報を伝播先のビューに伝播する。ユーザの伝播元のビュー操作によりトリガーされた連動情報はどの画面のどのビューに伝播させるかは、画面間の連動候補リストに基づいて、ユーザにコントロールさせる。

各画面内のビューに対応コンテナの連携関係により、画面間の連動候補リスト、さらに画面間のビュー間の連動候補リストを作り出すことができる。画面内のビュー連動候補リストを作り出すように、コンテナの連携可否情報から、画面間の連動候補リストをまとめる。

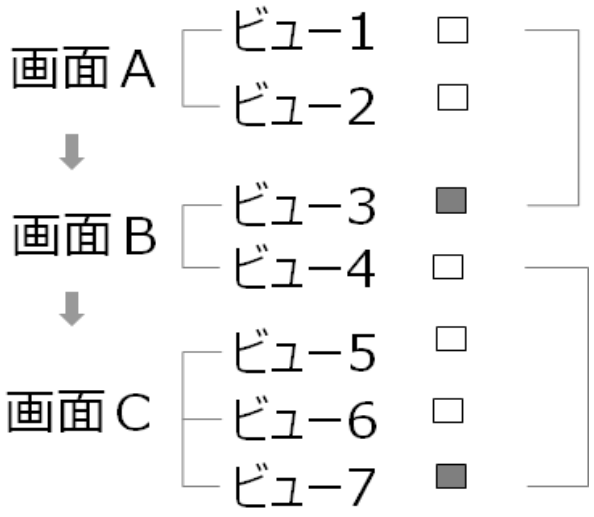

図１０　画面間の連動設定の例

図１０に示すように、画面Ａのビュー１→画面Ｂのビュー３、画面Ｂのビュー４→画面Ｃのビュー７に画面間連動フラグが設定されているから、画面Ａのビュー１の操作情報を画面Ｂのビュー３に伝播し、画面Ｂのビュー４の操作情報を画面Ｃのビュー７に伝播する。また、図９の赤い矢印が示すように、画面Ａのビュー１の操作を画面Ｂのビュー３へ、画面Ｂのビュー４の操作を画面Ｃのビュー７へ伝播される。画面間の連携は画面の配置するマシンに問わずに、ローカルマシンの複数画面間はもちろん、ローカルマシンと遠隔マシンの間にも連携できる。

実施

本提案のプロトタイプは、可視化プラットフォーム Polyspector™[3]をベースに構築した。検証のための構成は、図１１に示す。ノートパソコンを可視化端末として、１つの可視化端末を複数のダッシュボード（画面）を配置することができる。１つの可視化端末の複数の画面は、例えば、マルチ・ディスプレイや、ブラウザの複数のタブで画面を分けることができる。複数の可視化端末を用意すれば、遠隔画面間の連動になる。案１（図３）の画面間情報やり取りはサーバ経由で Websocket で実施した; 案２（図４）の画面間情報やり取りは WebRTC により実施した。両方ともデータ転送遅延がミリセカンドであるためインタラクテイブな操作の意味では省略できる。

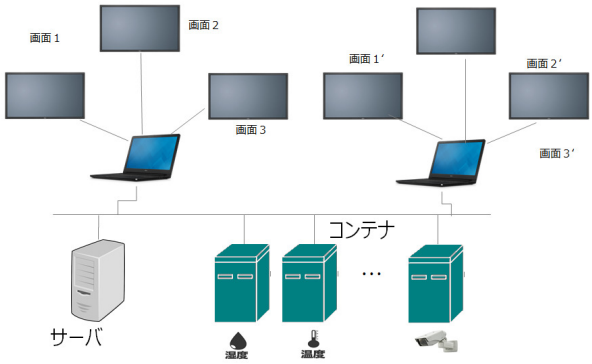

図１１　検証システム構成

コンテナ（データ処理を実行するプロセス）は独立の処理モジュールの総称である。Virtual Machine や Docker (https://www.docker.com)のコンテナ型の仮想化環境で構築できるが、本検証環境で簡易化するため、任意の複数のマシンで立ち上がった複数のプロセスで実現した。

コンテナ間、コンテナとサーバ間の通信は、RabbitMQ(https://www.rabbitmq.com/)を介して、言語非依存の通信環境を構築した。

図１２はビュー連携の画面例である。（２）→（１）、（１）→（２）、（３）→（４）のビュー連動が設定されていることがわかる。ビュー連動のデータ更新は、各ビューに対応しているコンテナ（データ処理モジュール）により行われる。図１２左側のビュー（１）とビュー（２）は、それぞれ異なるセンサーの時系列に対応している。それぞれの元データは、数年間の秒単位のサンプリングで、数千万～億のレコード数になっている。可視化端末は、それぞれのビューの幅（ピクセル）に対して、ピクセルごとのデータの統計値（最大値、最小値や平均など）を取っており、各ビューを生成するデータ数は数百～2000 個に、１回の描画時間が数百ミリセカンド以内に抑える。ビュー（１）とビュー（２）の操作連動は、コンテナ間にビューの更新した絞り込み範囲を共有し、ぞれぞれの元データからリサンプリングデータを計算し、ビューを更新する。裏側のコンテナの連携なしに、そのような大規模データのインタラクテイブなビュー連動は無理である。

参考文献：

[１] Li, X., Kuroda, A., Matsuzaki, H., Nakajima, N., Distributed Aggregate Computation between Server and Client for Interactive Visualization. 2015 IEEE 5th Symposium on Large Data Analysis and Visualization, pp. 135-136.

[２] Li, X., Kuroda, A., Matsuzaki, H., Nakajima, N.,

Advanced aggregate computation for large data visualization. 2015 IEEE 5th Symposium on Large Data Analysis and Visualization, pp. 137-138.

[3] ビッグデータ可視化プラットフォーム「Polyspector™」の開発について
(https://www.toshiba.co.jp/rdc/detail/1610_02.htm)

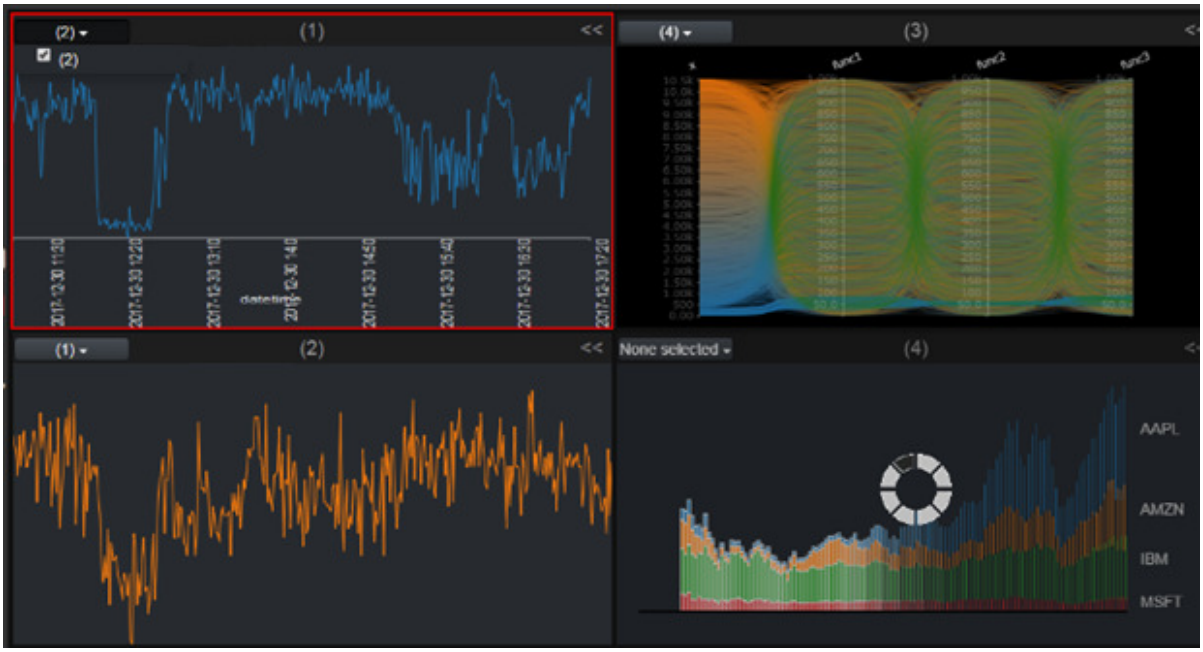

図 12 ビュー連携の画面例