Vol.2020-IOT-48 No.22
2020/3/3

動的適応性を持つ分散システムを対象としたシステム状態可視化手法の検討

林 友佳[1,a)]　伊勢田 蓮[1]　松原 克弥[1,b)]　鷲北 賢[2,c)]　坪内 佑樹[2]　松本 亮介[2]

概要：Web サービスの高機能化やニーズの多様化にともなって，Web サービスを構成するシステムが集中型から分散型に移行している．今後，IoT との連携により，システムの大規模化や超分散化が進むことが予測される．筆者らは，サービスへの負荷や要求などの状況に応じて，計算機資源の割当や配置を動的かつ柔軟に変更できるシステム基盤の実現を目指している．そのようなシステムでは，各コンポーネントの状態や依存関係を迅速に把握することが困難になる．そのため，障害発生時の原因追求や対処に時間を要してしまうという課題が起こりつつある．本研究は，動的に構成が変化する超分散システムを対象にした，システム監視ツールを実現する．本稿では，動的に変化するコンポーネントの位置や依存関係を迅速に把握するための可視化手法について検討する．

1. はじめに

SNS（Social Networking Service）や e コマース，オンラインゲームやメディア配信などといった，多種多様な Web サービスが普及するにともなって，それら Web サービスを実現するシステムが大規模化，複雑化している．さらに，スマートフォンの普及にともなうインターネットユーザの増加があいまって，Web サービスの利用者増加に対応して処理能力をスケールアウトできるシステム基盤が求められている．その結果，システムの内部構成は，モノリシックな集中型のアーキテクチャから，マイクロサービス・アーキテクチャに代表される分散型のアーキテクチャを採用する事例が増加している [1], [2]．今後，エッジコンピューティングなどの分散処理基盤技術の普及により，システムの超分散化がますます進むと予測される．

筆者らは，システムを構成する各コンポーネントを小規模のクラウド群へ分散配置し，対象サービスへのアクセス量や内部処理の負荷偏りなどの刻一刻と変化する状況に適応するために，それらへの計算資源割り当てや配置を動的かつ柔軟に変更していくことを可能にする，超個体型データセンターの実現を目指している [3]．そのような大規模かつ動的に構成が変化する分散システムでは，各コンポーネントの状態や依存関係を迅速に把握することが困難になる．そのため，障害発生時の原因追求や対処に時間を要してしまうという問題が起こることが考えられる．

本研究は，前述のような動的適用性を持つ分散システムを対象としたシステム監視技術の確立を目的としている．本稿では，動的適応性を持つ大規模分散システムを運用管理するうえで最も大きな課題のひとつとなる，複雑に依存関係を持つ多数のコンポーネントで構成されたシステム内で発生した障害の状況把握や発生要因の特定を支援するために，コンポーネント間の依存関係を俯瞰的に把握しつつ，各コンポーネントの状態変化を地理的・時間的に比較できる可視化手法について検討する．

2. 動的適応性を持つ分散システムの特徴と可視化で実現すべき支援

動的適応性を持つ分散システムは，ネットワークで接続された複数の計算ノード間にまたがってシステムを構成するコンポーネントが分散稼働し，サービスの状態，利用可能な計算資源状況，入出力データの位置やアクセス負荷などに変化に対して，構成や配置などの各コンポーネントの稼働状況を動的に変化させることで対処するシステムである．

動的適応性を持つ分散システムには，主に 3 つの特徴がある．1 つ目の特徴は，状態が頻繁に変化することである [4]．例えば，Web サービスへのアクセスが増え，特定のサーバに負荷がかかると，負荷を分散するために新たなサーバが立ち上がるということがある．そのため，リア

[1] 公立はこだて未来大学
Future University Hakodate
[2] さくらインターネット株式会社 さくらインターネット研究所
SAKURA internet Research Center, SAKURA internet Inc.
[a)] b1016016@fun.ac.jp
[b)] matsu@fun.ac.jp
[c)] north@sakura.ad.jp

ルタイムな情報を把握することが重要である．2 つ目の特徴は，依存関係が複雑であることである．分散システムでは，処理を分担する各コンポーネントが地理的に分散した場所で動作しつつ，ネットワーク通信を介して互いの入出力を連携させている．さらに，動的適応性を持つ分散システムでは，前述の状態変化に合わせて，動作するコンポーネントの数やそれらの依存関係も頻繁に変化する可能性がある．したがって，システム稼働中に変化する依存関係を常に把握することは難しい [5]．特に，このような分散システムでは，一部のコンポーネントに何らかの理由で異常が起こると，依存関係のある他のコンポーネントに影響を与え，その結果として大量のアラートが発生することがある．クリスマスツリー現象と呼ばれるこのアラートは，システム管理者に疲労を与えたり思考を阻害したりすることで，システムの状態や異常の原因を把握することが困難になり，障害への対応が遅れてしまう原因となる．したがって，分散システムでは，障害の根本原因を追求するために，依存関係を把握することが最も重要となる．3 つ目の特徴は，大規模であることである．動的適応性を持つ分散システムで採用されるマイクロサービスアーキテクチャでは，個々のコンポーネント（マイクロサービス）はそれぞれ独立しており，個別に開発やデプロイが可能である．そのため，コンポーネントごとにチームを組んで開発することがある．その結果，それぞれのチームが，特定のコンポーネントに対して専門的な知識を持つ一方，システム全体を正しく把握することは難しくなる．したがって，1 人や 1 つのチームではなく，それぞれのチームが協力することでシステム全体の監視を行うべきである．

以上 3 つの特徴を踏まえ，動的適応性を持つ分散システムの可視化では，ユーザの 3 つの作業を支援することを重視する．第一に，システムの状態の変化を素早く認識することである．第二に，システムのコンポーネントの依存関係を把握することである．第三に，開発や運用保守のチーム内，チーム間など，複数人で情報を共有することである．これら 3 つの作業を支援することで，ユーザが動的適応性を持つ分散システムの状態を効率的に把握することができる．システムの状態を効率的に把握することによって，障害発生時の対応を円滑に進めることができると考える．

3. 動的適応性を持つ分散システムの状態可視化に必要な要件

本章では，前章で挙げた動的適応性を持つ分散システムの状態可視化で支援する内容を基に，可視化に必要な要件を検討する．要件の検討にあたって，SA（状況認識：Situation Awareness）や認知心理学の観点から，可視化手法に適応可能な原則などを取り上げる．

3.1 Situation Awareness の観点に基づいた要件

SA とは，一定の時間と空間の中で環境にある要素を認識し，その意味を理解し，近い将来の状況を予測することである [6]．Endsley は，認識，理解，予測の段階をそれぞれレベル 1，レベル 2，レベル 3 に分けてモデル化して扱っている [6]．レベル 1 からレベル 3 までの SA の流れが，次に起こすべき行動や意思を決定するために重要である．いずれかのレベルに関する力や必要な情報が欠けていると，正しい状況認識ができず，正しい意思決定ができない．SA の概念は，これまで航空機，航空交通管制，製造システムや原子力発電所などの大規模システムの操作，医療などの領域で研究され，活用されてきた．システム監視と障害対応に，SA から意思決定までのプロセスを当てはめると以下のようになる．

レベル 1 SA　一定の時間内で，アラートや監視ツールの可視化表示から，システムのどの部分にどのような変化が起きているかを認識する

レベル 2 SA　認識した情報を集め，因果関係を踏まえて整理することで，システム全体の現状を理解する

レベル 3 SA　対応をしないとシステムが今後どうなるか，また，どのような対応をするとどのような結果になるかについて，現状の理解に基づき，予測する

意思決定　予測を基に，実際に行う対応の内容を決定する

また，複数の個人が同じ目的を果たすため，チームとして協力して意思決定と行動を行うことがある．この場合は，Team SA というチーム全体の SA を考えることができる．Team SA は，全てのチームメンバーが自分の責任を果たすために必要な SA を所有している程度のことである．例えば，航空機のコックピットで、機長と副操縦士の両方が特定の情報を知らなければいけない場合がある．副操縦士はその情報を知っているが，機長はその情報を知らないという事態が起こると，Team SA は悪影響を受けているといえる．この事態が解決しないと，その後の航空機の操作や 2 人の連携の質も低下する可能性がある．先に述べた通り，大規模かつ複雑なシステムは 1 人で全容を把握することが難しく，開発，運用保守チーム全体で協力して障害対応にあたるべきである．よって，通常の SA だけでなく Team SA も含めた観点から，動的適応性を持つ分散システムの状態可視化に必要な要件を考える．SA や Team SA の観点を取り入れた設計を SAOD（Situation Awareness Oriented Design）と呼ぶ．Endsley らは，システムの操作者の SA を支援し強化するために重要な，SAOD に関わる 50 の設計法則を提案している [7]．50 の設計法則のうち，システム状態の可視化に活用できる法則を以下に抜粋する．

a. Organize information around goals
情報は，情報を作成したセンサーやシステムに基づいて表示される技術志向の方法で提示するのではなく，ユーザの主要な目的に基づいて編成する．

b. Present Level 2 information directly - support comprehension
ユーザの注意やワーキングメモリは限られているため，レベル 2 SA の要件に基づいて予め処理，統合した情報を提供する．

c. Provide assistance for Level 3 SA projections
過去の値の変化や，情報から予測できる内容を表示することで，ユーザが今後のシステムの状態や周囲の環境を予測できるように支援する．

d. Support global SA
注意が絞り込まれすぎないよう，ユーザの目的に対する全体の概要を常に提供する．

e. Use information filtering carefully
SA に不必要な情報や，未処理のデータの削減は有益であるが，全体像の把握に必要な情報や，予測に必要な情報，必要性の有無に個人差があるような情報の削減は注意する．

f. Group information based on Level 2 and 3 SA requirements and goals
ユーザが目的達成に必要な SA を導出する際に，複数のシステムから横断的に探す必要がないように，同じ目的をサポートするために使われる情報は同じ表示領域に集中させる．

g. Minimize task complexity
ユーザの認知負荷やミスを軽減するため，必要なタスクを実行するために必要なアクションの数とアクションの複雑さを最小限に抑える．

h. Build a common picture to support team operations
各チームメンバーに提供される正確な状況は，それぞれのニーズに基づいて調整する必要があるが，判断の不一致が発生しないように共通のイメージを持つための情報は提供する．

i. Avoid display overload in shared displays
Team SA の強化に用いる共有ディスプレイで，共有する必要のない情報が大量にならないように注意する．

j. Support transmission of different comprehensions and projections across teams
他のチームメンバーの視点やレベル 2 SA を知ることができる機能によって，チーム内で異なっている理解や予測を共有することをサポートする．

以上のような法則を踏まえて導出した要件は，以下の通りである．

要件 1 特定の瞬間のデータだけではなく，前後のデータも表示されている（c）
頻繁に変化するシステムの状態の理解と今後の予測に役立てるため

要件 2 情報はグループ化や適切なフィルタリングなどがされた状態で表示されている（a，b，e，f，i）
認知的負荷を軽減した上で，頻繁に変化するシステムの状態や依存関係を把握，理解できるようにするため

要件 3 概要を常に提供していて，情報が過度に絞り込まれていない（d，e）
特定のコンポーネントからの情報だけでは把握できない，依存関係が複雑なシステムの状態の理解を支援するため

要件 4 開発，運用保守チームのメンバー間で，必要な SA は共有できるように支援されている（h，i，j）
大規模なシステムに関わる必要な情報共有を，円滑に行えるよう支援するため

要件 5 システムの状態確認や依存関係の把握，情報共有のために必要な操作は複雑すぎない（g）
ユーザの認知的な負荷やミスを軽減するため

3.2 認知心理学的観点に基づいた要件の具体化

具体的な可視化手法を検討するためには，SAOD より導出した要件の他に，ユーザインタフェースのデザインについても検討する必要がある．システムの状態可視化において，一般的に用いられるユーザインタフェースの画面はダッシュボードである．そこで，認知心理学的観点からダッシュボードのデザインについて検討する．

システム監視ツールにおけるダッシュボードとは，複数の情報源から集めたデータを，表やグラフなどの形に加工し，一覧できるようにした画面のことである．ダッシュボードは，システム監視以外にもビジネスの指標を示す際などに用いられている．Few は，ダッシュボードの設計で一般的な 13 の間違いや，ダッシュボードの設計をする際に気をつけるべき点，活用すべきデザインの法則について述べている [8]．書籍の中で事例として挙げられているダッシュボードは，ビジネス目的のものが多くを占める．しかし，デザインの指針としては，システム監視にも活用できると考える．ダッシュボードの設計で一般的な 13 の間違いのうち，監視ツール提供側が考慮すべき内容を抜粋したものは以下の通りである．

i. Exceeding the boundaries of a single screen
ユーザが一目で見ることができる範囲内に重要な情報がない．また，ユーザが比較したい情報が断片化されていて，同時に表示できない．

ii. Choosing a deficient measure
2 つの値を相対的に示すべきにも関わらず絶対値を軸に用いるなど，不十分な尺度を選択している．

iii. Arranging the data poorly
最も重要なデータが目立つようになっていないなど，データの整理が不十分である．

iv. Highlighting important data ineffectively or not at all
重要なデータを非効果的に強調するか，全く強調していない．多くのデータを目立たせようとした結果，乱

雑になり却って目立たなくなっている．

活用すべきデザインの法則は，ゲシュタルトの法則が挙げられている．視覚におけるゲシュタルトの法則（Gestalt Principles of Visual Perception）とは，オブジェクトを一緒にグループ化する傾向がある視覚的特性をまとめたものである．書籍では，以下の 6 つについて解説されている．

I. The Principle of Proximity
近くにあるオブジェクト同士を同じグループとして扱う傾向がある．

II. The Principle of Similarity
似た色，大きさ，形，方向のオブジェクトを同じグループとして扱う傾向がある．

III. The Principle of Enclosure
線で囲む，背景に色を付けるなどによって，境界があるとその境界ごとにグループとして扱う傾向がある．

IV. The Principle of Closure
完全に閉じられていない形でも，閉じられた規則的な形として認識する傾向がある．

V. The Principle of Continuity
インデントが揃っているオブジェクトなど，連続性があるように見えるオブジェクトを同じグループとして扱う傾向がある

VI. The Principle of Connection
線などで繋がれているオブジェクトを同じグループとして扱う傾向がある．I，II より強く働き，III よりは弱い働きを持つ．

書籍では，その他にもダッシュボードや画面全体のデザインをするときに気をつけるべきことを示している．内容のうち，システム状態の可視化に活用できるものを以下に抜粋する．

A. 強調したい要素は色，位置，形，動きで他と違いをつける
B. 最も強調される位置である画面上の左上や中央に，強調したい要素を配置する
C. グラフを結合する，比較したい要素同士を近くに配置する，比較したい要素同士に共通の色を用いてリンクさせる，比較用の値を表示する，などの工夫により，ユーザがデータを比較できるようサポートする
D. ユーザがデータと直接インタラクションすることで，追加データにアクセスできるようにする．ボタンなど個別のコントロールを排除する．

以上のような法則を基に導出した要件は，以下の通りである．

要件 6　ユーザにとって重要な情報は，スクロールや画面遷移が必要ない画面上の範囲に全て含まれている（i，B）
ユーザが重要な情報を見落とさないようにするため

要件 7　ボタンなどではなく，表示されているデータに何かしら操作をすることで，データの詳細などが表示できるようになっている（D）
ユーザが直感的に，ミスなく追加データにアクセスできるようにするため

要件 8　強調したい要素は，他の要素とは異なる色，形，動きなどをつけることで他の要素と区別されている（iii，iv，A）
ユーザが，システムの状態の変化を見落とさないようにするため

要件 9　一貫性を持たせたい要素同士は，同じ色，形，動きなどで構成されている（II）
ユーザの認識に不要なノイズを与えないようにするため

要件 10　意味のある比較を支援している（i，ii，C）
特定のコンポーネントに異常が発生した際，そのコンポーネントの過去の状態や，依存関係がある別のコンポーネントと比較できるようにし，現状の理解に役立てるため

要件 11　グループとして認識させたい要素は，ゲシュタルトの法則に基づいてグループ化されている（I〜VI）
ユーザに与える認知的負荷を軽減し，依存関係の可視化やグラフ表示を実現するため

4. 既存の監視ツールの分析

第 3 章で述べた要件を基に，既存のオープンソースソフトウェア監視ツールが，動的適応性を持つ分散システムの状態可視化に適している点と適していない点を分析する．本稿では，Zabbix，Grafana，Weave Scope の 3 種の監視ツールを分析する．

4.1 Zabbix

4.1.1 Zabbix の可視化手法に関する特徴

Zabbix は主に，Dashboard（**図 1**），Screens，Maps という画面でシステムに関する情報を可視化している [9]．

最初に表示される Dashboard は，widget という部品で構成される画面である．それぞれの widget には，システムの状態や発生した障害の内容などが可視化されている．Screens には，ユーザが設定したグラフなどがシステム全体やホスト別に表示される．Screens に表示されるグラフによって，システムの状態の推移を知ることができる．また，Screens は複数画面作成できる．複数の Screens は，Slide shows という機能を用いることで，指定した時間ごとに自動的に切り替えて表示できる．Maps は，ネットワーク構成の可視化に用いられる．背景に画像を設定し，その上にホストなどの elements を配置できるため，物理的なホストの位置関係などを示すことができる．障害が起こると，elements の背景の色とアイコン同士を繋いでいる link の色が赤色などに変わる．Maps 上の elements をクリックす

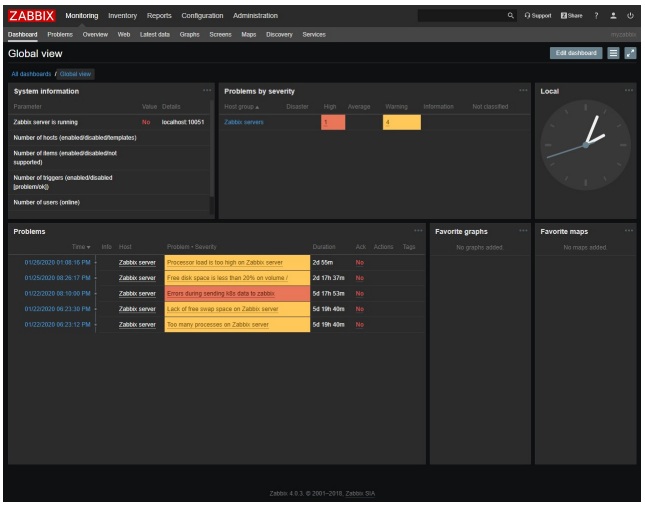

図 1　Zabbix の Dashboard 画面

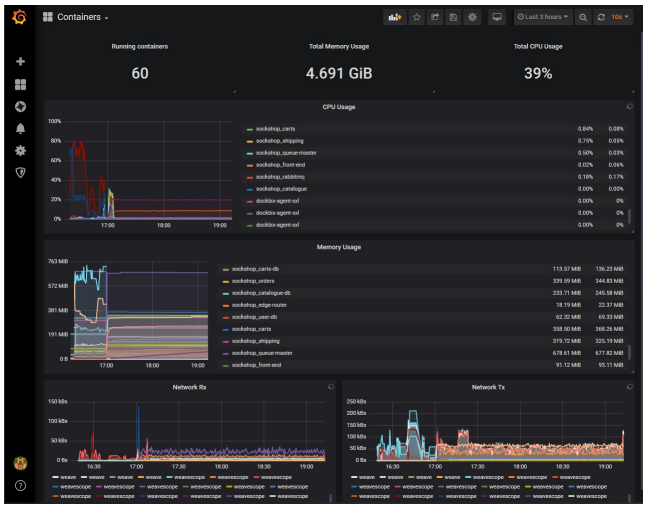

図 2　Grafana の Dashboard 画面

ることで，そのホストの状態を表示するグラフにアクセスできる．それぞれの画面では，kiosk mode というモードに切り替えることが可能である．kiosk mode は，操作メニューを非表示にし，各画面のデータの表示領域を広げるモードである．

4.1.2　Zabbix の可視化手法に関する分析

Zabbix が十分に満たすことができていない要件は以下の通りである．

要件 3　Screens のグラフによって，設定された監視対象のデータは表示することができる．Maps ではネットワーク構成を可視化している．リアルタイムに項目が追加されないため，動的適応性を持つ分散システムのリアルタイムな情報を提供しきれない可能性がある．

要件 4　障害対応時に記録としてコメントを残すことができる．グラフの特定の位置と紐付けたり，他のチームとリアルタイムに連携したりといったことはできない．

要件 6　Dashboard や Screens 上の配置はユーザが自由に編集することができるため，ユーザ自身が重要な情報を画面上部に配置することで達成できる．画面最上部にはメニューがあり，システムに関する表示領域は少し狭まるが，kiosk mode によって表示領域を広げることができる．しかし，メニューは固定されていないため，ページの下部を表示している状態から画面の切り替えなどを行うためには，スクロールが必要になる．

要件 9　Screens 上の複数のグラフで同じ監視対象がある場合，同じ色が使われている．監視対象が増えると色の種類も増え，一目で見分けることが難しくなる．

要件 10　複数の監視対象の同じ項目を同じグラフに統合して表示することで，比較を支援している．依存関係があるもののみを一時的に取り出すことはできないため，比較する意味のない監視対象同士が同じグラフにある可能性がある．

要件 11　Dashboard 上の widget はゲシュタルトの法則 III によって，適切にグループ化されている．Screens 上ではグラフ同士の間隔が狭く，線で囲われていないため，一見するとグループがわかりづらい．

4.2　Grafana

4.2.1　Grafana の可視化手法に関する特徴

Grafana は，Dashboard（図 2）という画面でシステムに関する情報を可視化する [10]．

Dashboard には，ユーザが設定したグラフなどが表示される．全てのグラフをユーザ自身が設定することも可能であるが，テンプレートが配布されているため，テンプレートを用いることも可能である．グラフにマウスポインタを合わせると，その点でのグラフの y 軸の値がポップアップされるとともに，x 軸の値がわかるように縦線が表示される．縦線は，マウスポインタを載せたグラフだけでなく、ダッシュボード上の他のグラフにも連動して表示される．また，複数の監視項目が含まれるグラフから，一部分の項目のみ表示するよう容易に切り替えることができる．さらに，グラフの凡例をクリックすることで，色を選択するポップアップが出てくる．これにより，画面を切り替えずにグラフに用いる色を変更できる．

また，Zabbix と類似した可視化機能もある．playlist は，Zabbix のスライドショーとほぼ同じ機能をもち，Dashboard を指定した時間ごとに自動的に切り替えて表示する．kiosk mode は Zabbix の同名のモードとほぼ同じ機能を持つ．Dashboard や playlist を大きなモニタで，情報の領域を広げて表示するために使用する．

4.2.2　Grafana の可視化手法に関する分析

Grafana が満たしていない要件は以下の通りである．

要件 3　設定された監視対象のデータは表示することができる．しかし，依存関係の可視化をする方法はない．

要件 5　Dashboard のテンプレートが配布されているため，状態確認は容易である．また，前項目で述べたとおり，

annotation や playlist，kiosk mode など，情報共有の操作も容易である．依存関係の可視化方法はない．

要件 9 Dashboard 上の複数のグラフで同じ監視対象がある場合，同じ色が使われている．しかし，監視対象が増えると色の種類も増え，一目で見分けることが難しくなる．

4.3 Weave Scope

4.3.1 Weave Scope の可視化手法に関する特徴

Weave Scope では，Graph（**図 3**），Table，Resources という画面でシステムに関する情報を可視化する．それぞれの画面で可視化する内容は，Processes，Containers，Hosts の中から選択できる [11].

Graph 画面では，プロセス，コンテナ，ホストの依存関係を可視化する．依存関係の変化に応じて，プロセス，コンテナ，ホストを表す nodes の位置やそれぞれの接続を表す線の位置関係がリアルタイムに変化する．nodes の下にはホスト名などが記載されている．CPU，Memory というボタンを切り替えると，Graph 画面の nodes 上に表示される情報が変わる．例えば，CPU を選択すると，nodes が現在の CPU 使用率に応じた面積分だけ塗りつぶされる．さらに，nodes にマウスポインタを合わせると，対象のアイコンと依存関係のあるプロセス，コンテナ，ホストの CPU 使用率の数値が nodes 上に表示され，nodes 同士を繋ぐ線上に，データの流れを示す矢印が表示される．nodes をクリックすると，ウィンドウの右側に，対象のプロセス，コンテナ，ホストに関する情報が載っているポップアップウィンドウが表示される．左側には対象の nodes と，それと依存関係のある nodes が表示される．Table 画面では，プロセス，コンテナ，ホストに関する情報を表形式で可視化している．例えば，コンテナを選択した場合は，コンテナの名前やどのくらい前に作られたか，IP アドレスや状態，CPU 使用率やメモリ使用量が表示される．それぞれの項目の値に応じて行の並べ替えをすることができる．表の列の順番は並び替えることができない．表をクリックすると，Graph 画面で nodes をクリックしたときと同様に，対象のプロセス，コンテナ，ホストの情報が表示される．Resources 画面では，プロセス，コンテナ，ホストの CPU 使用率，メモリ使用量を長方形の図を塗りつぶす面積によって表現している．

4.3.2 Weave Scope の可視化手法に関する分析

Weave Scope が満たしていない要件は以下の通りである．

要件 1 値の推移が表示されているのは，nodes をクリックしたあとに出てくるウィンドウ上の CPU 使用率，メモリ使用量のみである．他は，特定の瞬間の状態を表示している．

要件 2 関係のある nodes は同じ色が使われていて，依存関係はネットワークを示す線で判断できるため，グループ化はされている．CPU 使用率とメモリ使用量のどちらかしか表示できないのは適切なフィルタリングではない．

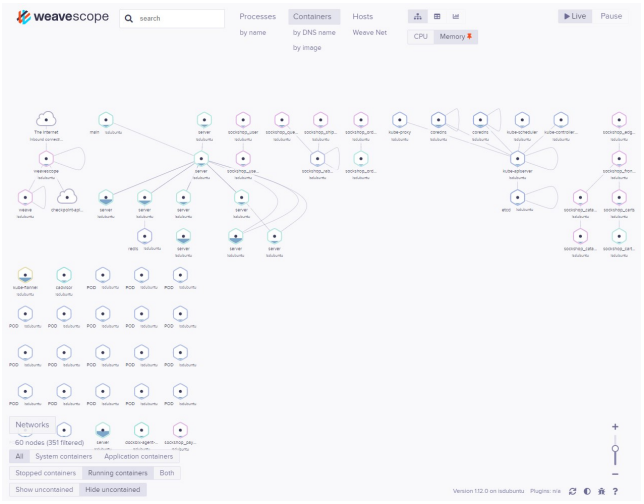

図 3 Weave Scope の Graph 画面

要件 3 基本的にはシステム全体の依存関係を常にリアルタイムで表示することができる．しかし，1 つの nodes の詳細を見ようとすると，その nodes の詳細情報とその nodes と依存関係のある nodes の表示のみに絞られてしまう．

要件 4 情報共有のためのコメント機能などは備わっていない．

要件 10 グラフなどで比較表示をすることができない．そのため，過去の状態との比較や，依存関係のある nodes との状態の比較が支援されていない．

4.4 既存の監視ツールの課題

4.1 から 4.3 の内容より，既存の監視ツールで動的適応性を持つ分散システムの状態を可視化するには，3 つの課題があるといえる．1 つ目は，複数のコンポーネントの状態変化と依存関係の可視化を関連させて同時に表示する手段がないということである．Weave Scope では，複数のコンポーネントの状態変化が表示できず，Grafana では，依存関係の可視化ができない．Zabbix では，それぞれを関連させて表示することができない．2 つ目は，監視対象が数百，数千という状態の時に適した可視化手法ではないことである．3 つの監視ツールでは，グラフ上で監視対象 1 つや 1 グループごとに表示色を分けたり，周りに監視対象の名前を表示したりすることで，監視対象を区別している．しかし，この方法では監視対象が増えると，ユーザが色や名前を見分けることが難しくなるため，区別のための情報を表示する意味がなくなってしまう．3 つ目は，チーム内，チーム間での情報共有手段に乏しいということである．Zabbix では，障害対応に関するコメントしか残すことができず，Weave Scope ではコメントやアノテーションを残す機能が

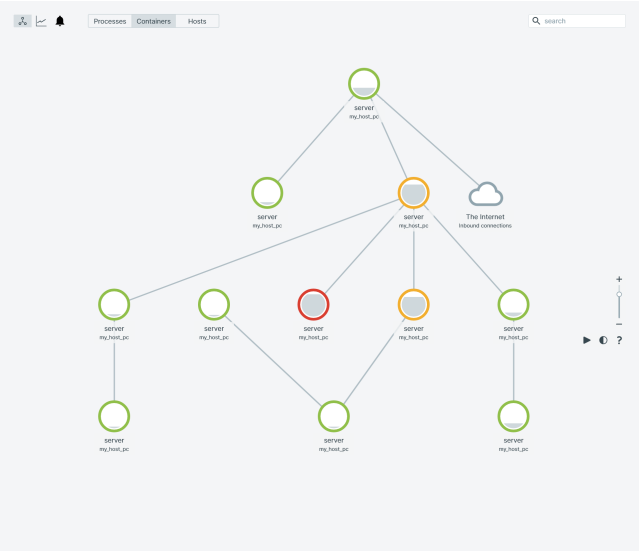

図 4　提案する可視化手法の画面

図 5　複数のコンポーネントをクリックした際の画面

ない．

5.　可視化手法の提案

本章では，第 4 章で挙げた既存の監視ツールの分析内容に基づき，動的適応性を持つ分散システムの状態可視化に適した可視化手法を提案する．

5.1　提案する可視化手法の概要と特徴

本稿では，コンポーネント間の依存関係を俯瞰的に把握しつつ，各コンポーネントの状態変化を地理的，時間的に比較できる可視化手法を提案する．地理的な状態変化とは，コンポーネント間の依存関係の動的な変化や，コンポーネントの置かれる状況の変化を指す．時間的な状態変化とは，各コンポーネントに関する定量的な値の変化を指す．提案する可視化手法の特徴は主に 3 つある．1 つ目の特徴は，複数のコンポーネントの地理的な状態変化と時間的な状態変化の可視化を関連させて同時に表示することである．図 4 のように，コンポーネントの依存関係の可視化を基本とする画面を作成することで，地理的な状態変化を可視化する．加えて，コンポーネントをクリックすることで，依存関係の表示をしたまま，それと関連させてコンポーネントの時間的な状態変化に関する情報を折れ線グラフで可視化する（図 5，図 6）．これにより，システムの状態の変化を素早く認識することと，システムのコンポーネントの依存関係を把握することを支援することができる．2 つ目の特徴は，ユーザが必要としている情報に合わせた可視化を行うことである．基本的には，地理的変化や時間的変化が頻繁に起こるコンポーネントが，重点的に監視する必要のあるコンポーネントだと捉え，画面上部に配置する．その他，ユーザの要求に合わせて，ユーザが指定した特定のコンポーネントを上部に配置した可視化や，フィルタリングを行い一部のコンポーネントのみを可視化することも可能である．これにより，ユーザにかかる認知的負荷を軽減し，システムの状態の変化を素早く認識することを支援する．3 つ目の特徴は，コンポーネントやグラフの特定範囲に紐付けてコメントができることである．コメント機能を示すアイコンをクリックし，それからコメントを残す対象を選択することで，コメントを残すことができる．これにより，開発や運用保守のチーム内，チーム間など，複数人で情報を共有することを支援する．

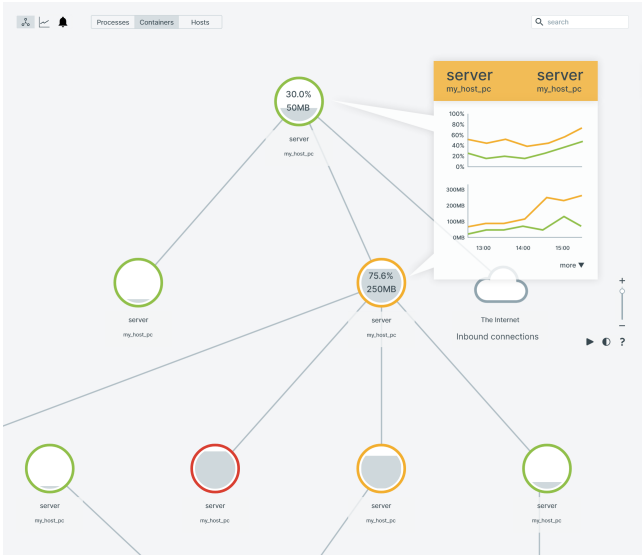

図 6　複数のコンポーネントのグラフを統合した際の画面

5.2　提案する可視化手法の分析

以下では，提案する可視化手法が第 3 章で述べた要件を踏まえて考案されていることを示す．

要件 1　コンポーネントの時間的な状態変化に関する情報を折れ線グラフで可視化できる．

要件 2　ダッシュボード画面上のグラフは，ゲシュタルト

の法則 III に則りグループ化している．また，複数の監視対象から一部の対象のみの表示に切り替えることができる．

要件 3 コンポーネントをクリックした際も依存関係の表示を維持できる．

要件 4 コンポーネントを示すアイコンや状態変化を示すグラフにコメントを付けることができる．

要件 5 システムの状態確認，依存関係の把握，情報共有に必要な手順は少なくなるように設計している．

要件 6 見るためにスクロールの必要がない画面上部には，重要なコンポーネントが表示される．また，操作で使うボタンは位置を固定しているため，常に画面上の範囲に含まれている．

要件 7 コンポーネントにマウスポインタを合わせる/クリックすることで，そのコンポーネントの詳細情報が表示できる．

要件 8 ユーザが設定した異常の基準に届いているコンポーネントは黄色，正常に動作していないコンポーネントは赤色の枠で表示されるため，通常状態のコンポーネントや背景，操作ボタンなどとは区別されている．

要件 9 データベースコンテナ同士やアプリケーションコンテナ同士など，似た役割を持つコンポーネント同士が一貫性のある表示になるよう，コンポーネントのアイコンの種類が使い分けられている．

要件 10 依存関係の可視化画面で，コンポーネントを複数クリックした際，表示されるグラフ同士はドラッグで統合できる．依存関係のあるコンポーネントのデータの比較が容易に，直感的に行える．

要件 11 コンポーネント間を繋ぐ線が，ゲシュタルトの法則 VI を活用していて，依存関係のあるコンポーネント同士をグループとして認識させている．

以上より，提案する可視化手法は要件を満たしているといえる．

6. まとめと今後の課題

本稿では，動的適応性を持つ分散システムを対象としたシステム監視ツールの実現を目指して，コンポーネント間の依存関係を俯瞰的に把握しつつ各コンポーネント状態の比較が容易な可視化手法について検討を行った．動的適用性を持つ分散システムの特徴を整理して，依存関係を含む状態変化の把握と情報共有を本システム監視に必須な機能要件と定義した．また，Situation Awareness と認知心理学の観点からシステム監視ツールに必要な要件を定義し，動的適応性を持つ分散システムを対象とした際の既存システム監視ツールに存在する課題を抽出した．これらの要件と既存監視ツールの分析から得た知見から，コンポーネント間の依存関係を示すグラフと，そのグラフへのインタラクションによって各コンポーネントの状態把握と比較が可能な可視化インタフェース提案した．

今後の課題として，数百から数千のコンポーネント数に対応するための依存関係の可視化手法について検討がある．大規模分散システムの依存関係を俯瞰的に把握するためには，一度に表示できる情報量を増やすことができる Tiled Display Wall のような大規模可視化装置の活用を考えている．また，現在のシステム監視において広く用いられている通知（アラート）機能について，大規模分散システムにおける課題分析や機能要件の検討を行う．さらに，本提案の可視化手法を適用したシステム監視ツールの実装を行い，有効性について評価を実施したい．現在，実装手法の検討とともに，システム監視ツールを評価するための分散システム・テストベッドの構築を行っている．

参考文献

[1] Pahl, C. and Jamshidi, P.: Microservices: A Systematic Mapping Study, *Proceedings of the 6th International Conference on Cloud Computing and Services Science*, p. 137146 (2016).

[2] Rademacher, F., Sachweh, S. and Zndorf, A.: Differences between Model-Driven Development of Service-Oriented and Microservice Architecture, *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*, pp. 38–45 (2017).

[3] さくらインターネット研究所：超個体型データセンターを目指す当研究所のビジョン さくらインターネット研究所，`https://research.sakura.ad.jp/2019/02/22/concept-vision-2019/` (accessed January 30, 2020).

[4] Matsumoto, R., Kondo, U. and Kuribayashi, K.: FastContainer: A Homeostatic System Architecture High-Speed Adapting Execution Environment Changes, *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, IEEE Computer Society, pp. 270–275 (2019).

[5] 坪内佑樹，古川雅大，松本亮介：Transtracer: 分散システムにおける TCP/UDP 通信の終端点の監視によるプロセス間依存関係の自動追跡，インターネットと運用技術シンポジウム論文集，Vol. 2019, pp. 64–71 (2019).

[6] Endsley, M.: Toward a Theory of Situation Awareness in Dynamic Systems. Human Factors Journal, *Human Factors: The Journal of the Human Factors and Ergonomics Society*, Vol. 37, pp. 32–64 (1995).

[7] Endsley, M., Bolt, B. and G., J. D.: *Designing for Situation Awareness: An Approach to User-Centered Design*, Taylor & Francis (2003).

[8] Few, S.: *INFORMATION DASHBOARD DESIGN*, O'Reilly Media, Inc. (2006).

[9] LLC., Z.: Zabbix とは，`https://www.zabbix.com/jp/` (accessed January 30, 2020).

[10] Labs, G.: Grafana: The open observability platform — Grafana Labs, `https://grafana.com/` (accessed January 30, 2020).

[11] WEAVEWORKS: Weave Scope: Automatically Detect and Process Containers And Hosts, `https://www.weave.works/oss/scope/` (accessed January 30, 2020).