Kubernetes環境下におけるPod単位でのCPU使用量分析のグラフ自動生成

増田 和範[1]　杉本 一彦[2]　串田 高幸[1]

概要：Kubernetes によってコンテナを管理することができる．コンテナは Pod 内に配置される．限られたリソース (CPU，メモリ) を過不足なく使用するために，Pod に対してリソース使用量を分析する必要がある．サービス管理者はこの分析を行うためにリソースメトリクスを用いる．Python で使用できるライブラリを使用することで統計分析を行うことができる．しかし，データの前処理を行う必要があり分析結果のグラフ画像をレポート形式で使用する場合，グラフやフォントのサイズが合わず作成後に修正することでユーザの作業が増加してしまう課題がある．そこで本提案ソフトウェアでは，ユーザがデータの保存がされているデータベースへアクセスするための情報を設定ファイルに書き込むだけで，分析結果を Web 上で表示し，その画像ファイルを任意で取得することを可能とする．ユーザの作業はデータベースへのクエリに必要な情報を設定ファイルに記述するのみである．それ以降のリソースメトリクスデータの取得，データの前処理，データの分析，分析結果のグラフ化，Web ページ上でグラフの表示，これらの作業は Python を用いたプログラムで自動化の実装を行った．出力されるグラフは修正をすることなく，そのまま使用することができる．

1. はじめに

背景

東京工科大学クラウド・分散システム研究室紹介サイトは WordPress で作成されている．WordPress はコンテナとして Pod 内に構築されおり，それらの Pod は Kubernetes によって管理されている[*1]．

Kubernetes は 2014 年 6 月 Google Developer Forum にて発表されたオープンソースのコンテナマネージャーソフトウェアである [1]．Pod とは Kubernetes が管理できる最小単位である[*2]．コンテナはアプリケーションを仮想化する技術であり，クラウドアプリケーション管理に取り入れられている [2]．コンテナは Pod 内に配置されている．1Pod 内に 1 コンテナを配置するという考えは最も一般的な Kubernetes のユースケースである[*3]．よってこの原則に基づいてコンテナを管理している場合，Pod ごとに CPU を分析することは，コンテナごとの分析を行っているということになる．

WordPress は Web サイトの作成と管理を容易にするオープンソースのコンテンツ管理システムであり，世界で最も人気のある CMS である[*4]．CMS は Contents Management System の略称である．WordPress は Web サイト全体のシェアの 43.3%のシェア率であり，CMS のシェアだと 65.1%にも昇る[*5]．

Kubernetes で管理されているソフトウェアを監視する手法として Prometheus を利用したリソースメトリクス監視が挙げられる [3]．Prometheus は Pod 単位にリソースメトリクスを取得して，取得したリソースメトリクスを Grafana を利用することでグラフを作成できる[*6]．

上記の専用のアプリケーションを使用しない場合, Python によってリソースメトリクスを分析，グラフ化することができる．Python にはデータ分析に必要なライブラリ (Pandas, Numpy) やグラフ作成に必要なライブラリ (matplotlib) がオープンライブラリとしてある[*7]．これらを pip コマンドや conda コマンドでインポートすることでデータ分析を行い，結果を可視化することができる．

[1] 東京工科大学コンピュータサイエンス学部
〒 192–0982 東京都八王子市片倉町 1404-1
[2] 東京工科大学 大学院コンピュータサイエンス専攻
〒 192–0982 東京都八王子市片倉町 1404-1

[*1] https://drive.google.com/file/d/1atY8WHCxclRdF6qfAtV56vvf_P8Pq32n/view
[*2] https://cloud.google.com/learn/what-is-kubernetes
[*3] https://kubernetes.io/docs/concepts/workloads/pods
[*4] https://sitecare.com/blog/why-is-wordpress-so-popular
[*5] https://w3techs.com/technologies/segmentation/cl-ja-/content_management
[*6] https://kashionki38.hatenablog.com/entry/2020/08/06/011420
[*7] https://www.codexa.net/machine-learning-python-library/

課題

Python のデータ分析用のライブラリを使用することでリソースメトリクスを分析することができる．分析を行う前に図 1 のようにクラウド上のデータベースにリソースメトリクスが保存されている場合，データベース管理アプリケーションにクエリを送信し対象のデータを取得する必要がある．その際にクエリを作成する必要がある．

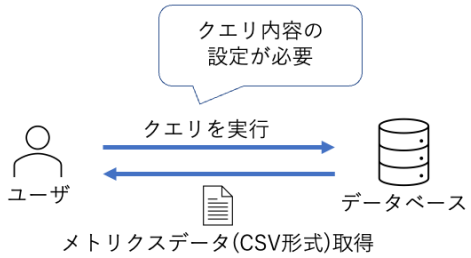

図 1　メトリクスデータの取得

そしてデータを取得後に図 2 で示しているように，データの分析後，分析結果をグラフや表として表示するまでに多くの作業工程が必要となる．この作業工程が増えてしまうことがリソースメトリクス分析における課題である．

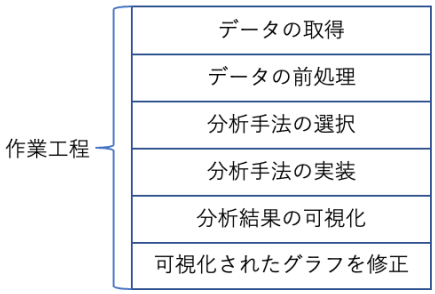

図 2　作業工程数内容の説明

各章の概要

第 2 章では, 本論文の関連研究を説明する. 第 3 章では, 本研究の課題を解決するための提案手法を説明する. 第 4 章では, 第 3 章の提案手法を実現するための実装を説明する. 第 5 章では, 第 3 章の提案手法の実験環境と実験の結果の分析を説明する．第 6 章では, 提案, 実験, 評価が本研究の課題を解決しているかを議論する.

2.　関連研究

関連研究として，Nicolas Chan らが行った A Resource Utilization Analytics Platform Using Grafana and Telegraf for the Savio Supercluster がある [4]．この論文ではリソースの使用率の統計はサービス管理者と利用者両方に利益が生じると述べている．そして統計結果を視覚的に表示することで、リソース使用傾向の特定や探索が可能になると述べている．この論文ではコンピュータの各ノードからリソースメトリクスを InfluxDB へ保存し，マスターノードからジョブデータを PostgreSQL へ保存し，Grafana でこれらのデータの可視化を行った．メトリクスに加えジョブデータを分析に利用することで，使用者はリソースの使用状況を分析することが可能となった．しかし，今回の分析で用いたのは限られた種類メトリクスであることが課題として挙げられる．そして Aidi Pi らが行った Profiling distributed systems in lightweight virtualized environments with logs and resource metrics がある [5]．この論文では複数の分散されているマシンの監視に対してメトリクス監視が有用であると述べられている．既存の手法ではログの解析を行うことでマシンごとの監視を行っていたが，リソースメトリクスを分析結果を可視化することで管理者がマシンの監視を行いやすくなると述べている．この論文では提案ソフトウェアの使用者は効率的に異常を発見し，その根本原因を突き止めることができ，実験結果から提案ソフトウェアは，ワークフローの理解，バグや干渉による異常の発見に役立つと述べられている．しかし，出力されるグラフを論文で利用しているが，フォントが小さくて認識できないことが課題として挙げられる．

3.　提案

提案方式

データベースに保存されているリソースメトリクスからデータ分析と分析結果のグラフ作成を行う Web アプリケーションを提案する．このアプリケーションを GitHub で公開し自分のコンピュータで行うことを想定する．今回は Pod ごとの CPU 使用量の時間推移を行う．

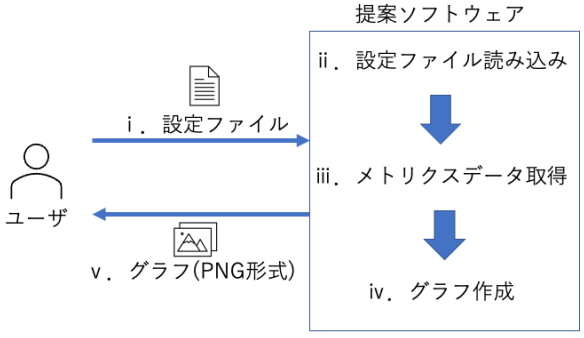

図 3　提案ソフトウェアの役割

図 3 における i から v について説明を行う．

i．ユーザはソフトウェアを実行する前に，設定ファイルを用意する必要がある．

ii．ソフトウェアを実行すると，i で作成した設定ファイ

ルを読み込む．そしてデータベース管理アプリケーションに送るために設定ファイルの内容をクエリへ変換する．

iii．作成されたクエリをクラウド上のデータベースへ送信し，リソースメトリクスデータを CSV ファイル形式で取得する．

iv．取得した CSV ファイルをもとに，分析を行い，結果を可視化するために，グラフを作成する．作成したグラフは PNG 形式で保存される．

v．作成したグラフ画像を Web 上で表示し，ユーザが任意でダウンロードできる．

ダウンロードされた画像は二段組のレポートにてサイズ指定をしなくても，そのままで使用することができる．

ユースケース・シナリオ

ユースケースを示す図 4 について説明を行う．

ブログサイト運用者が WordPress を使用して，ブログサイトを運用していることを想定とする．WordPress はコンテナ内に構築されている．コンテナは Kubernetes 上で Pod として管理されている．管理者は WordPress の各 Pod がノード内の CPU をどれだけ使用しているか，また，リクエスト数と CPU 使用量の関係性を分析することで今後の運用のためにノード内の CPU 使用量を効率的に使用したいと考えている．しかし，統計データを元に分析する場合，課題の章で述べたように，グラフを作成するまでに作業を多く要し，結果的に作業時間が増加してしまう．

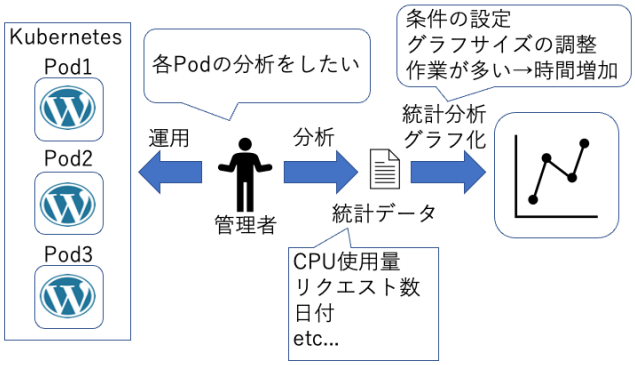

図 4　ユースケースシナリオ

そこで今回の提案ソフトウェアを利用することで，ユーザはリソースメトリクスデータを分析し可視化するまでに必要な作業をすることがなくなる．

4.　実装

今回の提案ソフトウェアは図 5 と図 6 で示した流れに沿って処理を実行する．

図 5 では app.py を実行後，Flask サーバが作成され，ユーザが初めにアクセスする Web サイトの top.html と分析結果のグラフを表示する show.html を作成する流れを示している．

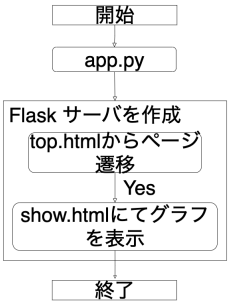

図 5　全体の流れ 1

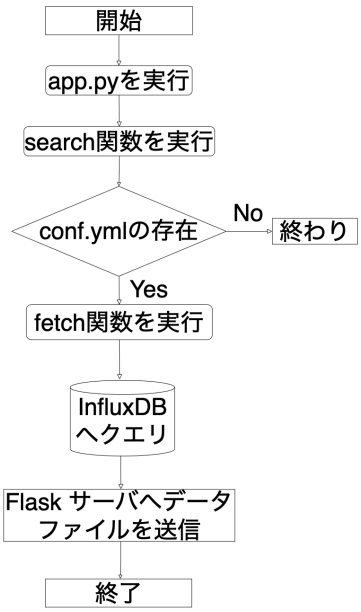

図 6　全体の流れ 2

図 6 では app.py を実行後，search 関数を実行し，conf.yml が用意されているかを確認する．そしてない場合はその時点でプログラムを強制終了し，ある場合は，fetch 関数によって conf.yml をもとに InfluxDB へクエリを行い，Flask サーバへクエリを元に作成されたリソースメトリクスのデータファイルを保存する流れを示している．ここの述べられた関数やファイルについて以降で説明を行う．

図 7 に提案ソフトウェア構成図を示す．

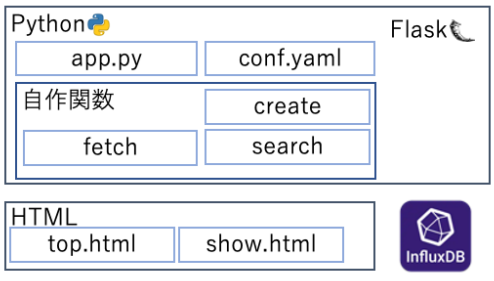

図 7　ソフトウェア構成図

図 7 で述べられている，プログラムファイルと今回作成した関数について説明を行う．

始めに Python で書かれたプログラムについて説明を行う．

app.py は提案ソフトウェアの根幹となる Python ファイルである．このプログラムを実行することで，CSV ファイルを取得するための fetch 関数や，分析結果のグラフを PNG 形式で保存する create 関数を実行することができる．そして保存されたファイルを Web サイト上で表示するために必要な Web アプリケーションサーバの作成を Flask を使用して実行する役割を担っている．

config.yml は InfluxDB へ送るクエリの情報を記述する設定ファイルである．config.yml への記述項目を以下に示す

- 対象データベース URL
- token
- org
- 対象期間
- namespace
- bucket

対象データベース URL はどのデータベースにクエリを行うかを決定するために必要である．次の token は API を用いて外部のアプリケーションからアクセスする際に，認証をするために使用する．org は対象のデータベースがどの組織が所有しているかを示している．対象期間は日付の形式を「YYYY-MM-DD HH:MM:SS」と指定し，対象期間の始まりと終わりを指定する．そしてデータの取得間隔も指定することができる．namespace は同一の物理クラスタ上で複数の仮想クラスタを構築するために用いる．よって単一物理クラスタ内の仮想クラスタを指定するために用いる．bucket はデータを格納するコンテナである．よってどの org が所有するデータベースの中からデータベースを特定するために用いる．

search 関数はサーバの指定ディレクトリ内に存在するかを探す．ない場合はユーザにコンソール画面で config.yml を作成するように促し，実行中の app.py を強制終了する．

fetch 関数は設定ファイルを元に InfluxDB へクエリを行い，データベースから該当データを CSV ファイル形式で取得することができる．この CSV ファイルは実行環境内に保存される．

create 関数はリソースメトリクスデータが保存されている CSV ファイルからグラフを作成し，Flask サーバ内に保存する．CSV ファイルのデータ形式をそのまま使用するのでなく，データの前処理を行い，Python ライブラリの matplotlib を使用して，データのグラフ化を行う．

top.html ファイルはユーザが Web アプリケーションへアクセスした際にはじめに表示されるページであり，このページには実行ボタンを配置し，このボタンを押すことで，show.html へページ遷移することができる．

show.html ファイルは分析結果のグラフを表示するページである．このページ上でユーザは任意でグラフを保存することができる．

5. 実験と分析

実験環境

図 8 で示している環境で実験を行った．

研究室紹介サイトの WordPress アプリケーションはコンテナ化され Pod として Kubernetes によって管理されている．Locust から研究室紹介サイトの WordPress サイトへリクエストを発生させる負荷試験を行う．Telegraf が Pod 単位でリソースメトリクス監視を行っている．生成されたリソースメトリクスは InfluxDB へ送られ，保存される．今回使用するリソースメトリクスは CPU 使用量である．

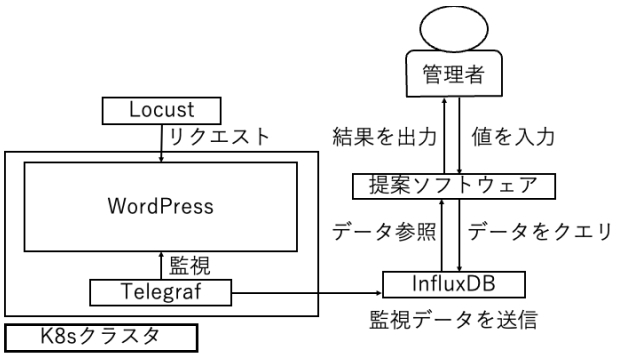

図 8 実験環境

実験結果と分析

提案で紹介した分析手法を用いた複数のグラフを Web サイト上で表示することができた．提案ソフトウェアの実行手順は以下の通りである．

(1) conf.yml へ必要情報の記述を行う．
(2) ターミナル上で以下のコマンドを行う．
    $python3 app.py
(3) 指定のポートへアクセスすると，図 9 が表示される．グラフ作成ボタンをクリックすることでページ遷移が行われる．
(4) ページ遷移後，図 10 が表示される．このページに表示されている．グラフ画像を任意で保存する．

はじめに，

グラフ作成時に必要な作業工程についての本提案と既存の比較を図 11 に表す．

図 11 のように，既存の方法では作業工程数が 6 に対し，提案ソフトウェアでは 4 に短縮することができる．工程数比較の結果を図 12 で示す．そして，提案ソフトウェアは conf.yml への書き込みをする必要があるが，それ以降の作

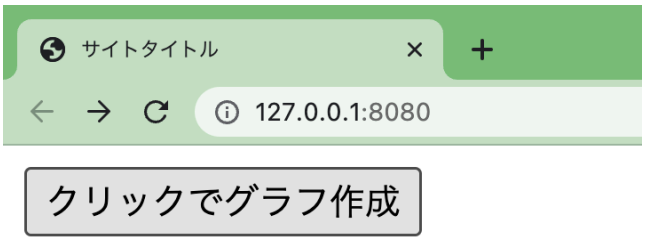

図 9　Web アプリケーションのトップページ

図 10　グラフ一覧表示ページ

業は既存の方法と比較してコードを記述する必要がないため，工程数が少なくなることに加え作業が容易になる．

既存の方法	提案ソフトウェア
データの取得	設定ファイルへ記述
データの前処理	提案ソフトウェアを実行
分析手法の選択	Webページへアクセス
分析手法の実装	グラフを任意で保存
分析結果の可視化	
可視化されたグラフを修正	

図 11　提案との比較

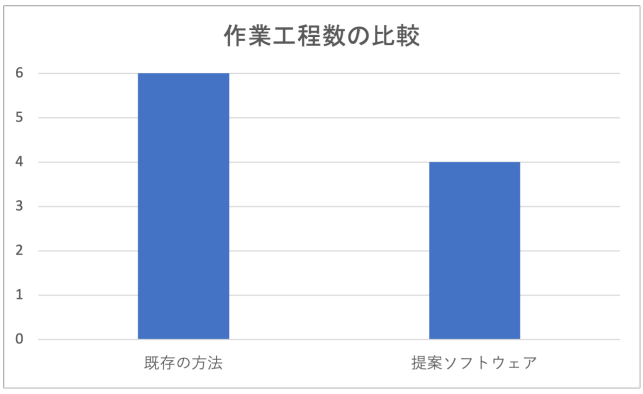

図 12　実験結果の比較

6. 議論

本提案ソフトウェアでは分析するリソースメトリクスが保存されているデータベースが InfluxDB に限定されているので，他のデータベース管理アプリケーションでも利用できるように設定ファイルと fetch 関数を拡張する必要がある．キャパシティプランニングを行うには，CPU 使用量の時間推移の分析だけではなく，リクエスト数と CPU 使用量の相関関係を分析する単一回帰分析を取り入れることで追加の分析手法を実装させる予定である．また同一 namespace 内に複数のサービスが混在している場合，Pod の分析だけではなく，サービス単位での分析を行う必要がある．

7. おわりに

本提案では，Kubernetes 上の Pod の CPU 使用状況を分析する際に発生する作業工程を減少させることを目的とした．CPU 使用状況の分析をソフトウェア内にある，自作した関数でデータの前処理，分析，分析結果をグラフ化，そして Web サイト上で表示することができた．これらの画像ファイル保存し二段組のレポート形式で使用する際に資料として利用しても本文とフォントサイズが変わらず視認できる．しかし，分析手法が CPU 使用量の時間推移での分析しか実装できていないため追加の分析手法を実装する必要がある．そのため今後は CPU 使用量とリクエスト数の関係を求めるために，単一回帰分析を実装を行う必要がある．

謝辞　本テクニカルレポートの執筆にあたりご助言を賜りました東京工科大学大学院バイオ・情報メディアコンピュータサイエンス専攻の大野 有樹さん，東京工科大学コンピュータサイエンス学部の平尾 真斗さんに感謝申し上げます．また実装にご協力いただきました東京工科大学大学院バイオ・情報メディアコンピュータサイエンス専攻の小山 智之さん，東京工科大学コンピュータサイエンス学部の坂本 一俊さんにお礼申し上げます．

参考文献

[1] Bernstein, D.: Containers and Cloud: From LXC to Docker to Kubernetes, *IEEE Cloud Computing*, Vol. 1, No. 3, pp. 81–84 (online), DOI: 10.1109/MCC.2014.51 (2014).

[2] Pahl, C., Brogi, A., Soldani, J. and Jamshidi, P.: Cloud Container Technologies: A State-of-the-Art Review, *IEEE Transactions on Cloud Computing*, Vol. 7, No. 3, pp. 677–692 (online), DOI: 10.1109/TCC.2017.2702586 (2019).

[3] Sukhija, N. and Bautista, E.: Towards a Framework for Monitoring and Analyzing High Performance Computing Environments Using Kubernetes and Prometheus, *2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Comput-*

ing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), pp. 257–262 (online), DOI: 10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00087 (2019).

[4] Chan, N.: A Resource Utilization Analytics Platform Using Grafana and Telegraf for the Savio Supercluster, *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (Learning)*, PEARC '19, New York, NY, USA, Association for Computing Machinery, (online), DOI: 10.1145/3332186.3333053 (2019).

[5] Pi, A., Chen, W., Zhou, X. and Ji, M.: Profiling Distributed Systems in Lightweight Virtualized Environments with Logs and Resource Metrics, *Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '18, New York, NY, USA, Association for Computing Machinery, p. 168–179 (online), DOI: 10.1145/3208040.3208044 (2018).

