KubernetesのPVの閾値を用いたNFSサーバーストレージ管理によるストレージ容量の超過を延期

三上 翔太[1]　圖齋 雄治[1]　串田 高幸[1]

概要：EC サイトの構築する際の手法として大きく，クラウドを使用する手法とオンプレミスを使用する手法の 2 種類に分かれる．オンプレミスで EC サイトを構築する際には一般的にオーケストレーションツールとして Kubernetes，ストレージを管理するために NFS サーバー (Network File System) を用いる．Kubernetes と NFS サーバーを用いて使用する際に Kubernetes の PV(PersistentVolume) が効力を持たないため，閾値に関係なく容量を増やすことができる．それはストレージ管理の観点からすると問題である．そのような問題を無視したまま EC サイトを運用することは NFS サーバー側にも支障をきたす．そこで Kubernetes の PV に効力を持たせ，PV を用いて NFS サーバーのストレージを管理する方法を提案する．その際，Pod の数に応じて均等に PV の閾値を割り当てると，1 つの Pod の PV が閾値に達した際に他の Pod の空き容量が多くなってしまう．そこで全ての Pod の増加量をもとに各 Pod の閾値を算出する．この提案手法を用いた結果，3 年の運用を想定とした EC サイトにおいて，1 つの Pod の PV が閾値に達した際，他の 3 つの Pod の PV の空きが合計 3.7GB となり，比較対象の 7.2GB と比較して 3.5GB 空き容量を減らし，約 50.6 %削減することができた．

1. はじめに

背景

近年，電子商取引の市場規模の拡大に伴い，EC サイトの数は増加している [1–3]．EC サイトの構築をする手法はクラウドを使用する手法とオンプレミスを使用する 2 種類に分かれる．大規模サイトはオンプレミス，中小規模サイトはクラウドで構築されることが多い[*1]．楽天グループ株式会社では，自社の EC サイトを自社サーバーで構築し，運営している[*2]．EC サイトの運用方法として，Kubernetes を用いる場合がある[*3]．

Kubernetes とはデプロイやスケーリングを自動化したり，コンテナ化されたアプリケーションを管理したりするための，オーケストレーションツールである [4–6][*4]．Kubernetes には PV(PersistentVolume) という仕組みと自動スケーリングという機能がある．PV とは，ストレージクラスを介して管理者または動的にプロビジョニングされるクラスターのストレージの一部であり，ノードと同じくクラスターリソースの一部である [7][*5]．自動スケーリングとはアプリケーションの負荷や需要に応じて Kubernetes クラスター内のリソースを自動的にスケールアップまたはスケールダウンする機能である [8,9]．

NFS サーバー (Network File System) とは，コンピュータがネットワーク上でファイルを共有するために開発された分散ファイルシステムである [10–12]．NFS サーバーを Kubernetes のストレージとして使用する場合，本来そのデータがあるコンピュータ上でのみアクセスできるファイルを，ネットワークを通して別のコンピュータからも共有したり読み込んだりすることが可能になる．よって，無駄なストレージの重複を避け，データの矛盾を防ぐことができる [13][*6]．

図 1 は NFS サーバーを Kubernetes のストレージとして使用する場合を示したものである．図 1 のように Kubernetes クラスターと NFS サーバーがネットワークを介して接続されている関係性であるため，Kubernetes の PV が不足しても NFS サーバーから参照できるようになっている．

EC サイトは新商品情報や商品の在庫状況，お客様の会員情報があげられ，それらを定期的に更新し，多くのユー

[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

[*1] https://ec-orange.jp/ec-media/?p=18512
[*2] https://xtech.nikkei.com/atcl/nxt/column/18/00001/07478/
[*3] https://cloud.google.com/blog/ja/topics/customers/google-kubernetes-engine-ec-cube
[*4] https://kubernetes.io/ja/
[*5] https://kubernetes.io/ja/docs/concepts/storage/persistent-volumes/
[*6] https://x.gd/ctITc

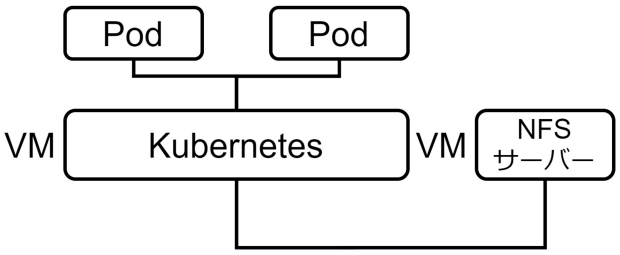

図 1 NFS サーバーを Kubernetes のストレージとして使用する場合

ザーに利用されている [14][*7]．しかし，運営においてデータが増加し続けることで，PV のストレージ容量が不足するという問題が発生する．PV のストレージ容量を増加すると，NFS サーバーストレージの容量が不足してしまい，NFS サーバーストレージを使用しているサービスにすべてに支障をきたしたり，データを損失してしまう可能性がある．

課題

課題は，Kubernetes と NFS サーバーを用いて使用する際に PV が効力を持たないことで PV で指定された容量を超えてもアプリケーションの運営においては問題が発生せず EC サイトの運営ができるため，NFS サーバーの枯渇に気づき辛い場合がある[*8]．EC サイトを運営しているとストレージが不足する場合がある．その理由として PV に本来の効力がないので EC サイトを構築した際に指定した容量を超えても管理者が気づきずらい場合があり，NFS サーバーのリソースを使い果たしてしまうこと挙げられる．

また，PV の実際の使用量と定義された容量を比べたときに実際の使用量が定義された値を上回っていたり，逆に使用率が少ないといったギャップがある．よって，管理者にとってストレージ管理が困難になる．

図 2 はストレージの使用量の変化を示したものである．NFS サーバーの使用を開始する際には管理者が余裕を持たせてストレージ容量の確保をする．よって最初は NFS サーバーストレージの容量にも余裕がある．しかし，数年 EC サイトの運営をすると少しずつ容量が埋まっていき，ストレージの空きが少なくなってしまう．[*9]

そこで PV に効力を持たせるとする．そうすると，PV が指定した閾値まで達した場合に Pod に現状以上のデータの書き込みができなくなり，アプリケーションのパフォーマンスに影響が出る．また，1 つのサービスが閾値に達したときに，図 3 の Pod B のように閾値よりもストレージ使用量が少ないとストレージの空きスペースが生まれてしまう．

[*7] https://www.onemarketing.jp/contents/bigdata-re/
[*8] https://qiita.com/MahoTakara/items/c07b1b3418fc9b7130f6
[*9] https://info.securesamba.com/media/10514/

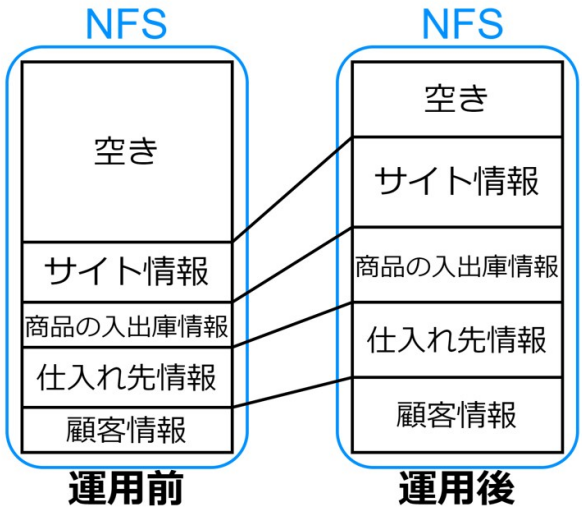

図 2 ストレージの使用状況

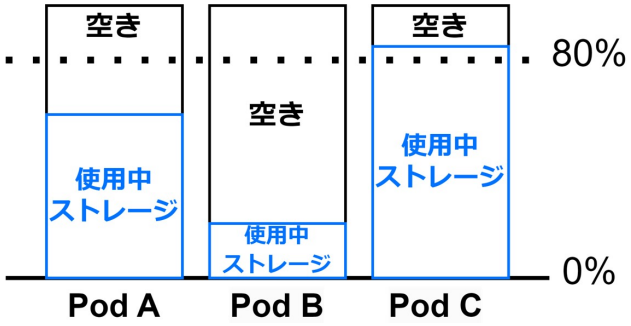

図 3 各 Pod のストレージ使用状況

各章の概要

第 2 章では関連研究について述べる．第 3 章では本稿の提案方式の説明とユースケースの説明する．第 4 章では本稿の提案に対する基礎実験の方法について述べる．第 5 章では評価方法と分析手法について述べる．第 6 章では本稿の議論についてを述べる．第 7 章では本稿のまとめを述べる．

2. 関連研究

Kubernetes におけるコンテナの永続ボリューム自動スケーリングについての研究がある [15]．この研究ではリソースの現状を把握し，リソースの増減を自動で行うアルゴリズムについて提案している．この論文で提案されている自動化によってリソースの節約ができていることを実証した．本稿ではこの論文で行っている増減の増加のみを行う．

物理マシンまたは仮想マシンの展開に基づくリソースの割り当て，およびコストの問題を考慮して，Kubernetes コンテナ配置テクノロジに基づくデータベースコンテナ化展開スキームが提案している研究がある [16]．この研究ではデータベースのコンテナ化展開スキームを行うことでデー

タベースコンテナインスタンスにおいて高性能ストレージが確保され，ストレージ使用率が向上した．この研究でのリソースは CPU も対象になっているが，本稿では CPU は対象とせず，ストレージのみとする．

仮想化テクノロジーを使用して，アプリケーションの需要に基づいてデータセンターのリソースを動的に割り当て，使用するサーバーの数を最適化することでグリーンコンピューティングをサポートするシステムについての研究がある [17]. この研究ではエネルギー節約と過負荷制御を実現するヒューリスティックが開発され，トレース駆動の実験結果で優れたパフォーマンスが実証された．この研究で「歪度」の概念を導入し，サーバーの多次元リソース使用の不均一性を測定した．本稿では「歪度」で見出すことができる非対称性のように実際の値との異なりを表せるようにする．

3. 提案

提案方式

本稿では Pod の PV のスケーリング可能な上限値を定めることで NFS サーバーストレージの超過を延期させ，サービスが停止したときのストレージの空き容量を減らす提案をする．本稿の提案では増加割合を使用するがサービスをデプロイしたタイミングでは増加割合を算出することができないので，式 1 により PV のスケーリング可能な上限値を算出する．

$$\mathbb{V} = \frac{TAS \times (1 - M)}{NS} \tag{1}$$

V　Volume(閾値)

TAS　Total Available Storage(総利用可能ストレージ)

NS　Number of Service(サービス数)

M　Margin(安全マージン)

TAS は総利用可能ストレージを指し，本稿の場合では NFS サーバーストレージの利用可能量を表す．その値を分母に用いられている 1 つの NFS サーバーでデータを管理しているサービス数で割ることで，各サービスを動かしている Pod の PV に同じ値が均等に割り振られることになる．

$$\mathbb{V} = (CSU + RNS \times GR) \times (1 - M) \tag{2}$$

V　Volume(閾値)

CSU　Current Storage Usage(現在のストレージ使用量)

RNS　Remaining NFS サーバー Storage
(NFS サーバーストレージの残りの使用可能量)

GR　Growth Rate(増加割合)

M　Margin(安全マージン)

式 1 により算出した PV の閾値にどれか 1 つの Pod が達した際に，式 2 によって閾値を決めなおす．決めなおした閾値に達した場合には式 2 による算出を再度行う．CSU は各 Pod のストレージ使用量を表している．RNS は閾値を決めなす段階での NFS サーバーストレージの残りの使用できる量を表している．GR は今までのストレージの増加量から導き出す増加割合を表す．CSU に RNS を GR に応じてプラスすることでストレージを多く使用してきた Pod の PV の閾値は大きくなり，逆に使用量の少なかった Pod の PV の閾値は小さくなる．式 1 で用いられている M(安全マージン) は管理者によって定める値とする．安全マージンを 10 %としたい場合は 0.1 となる．本稿では 20 %とするので値は 0.2 となる[*10].

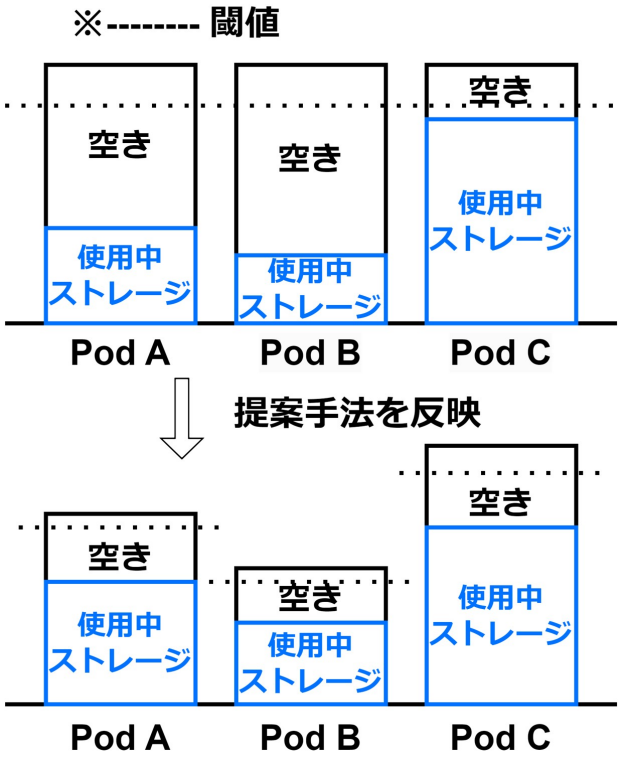

図 4　提案手法を反映させた例

図 4 は提案手法を反映させた例である．Pod C はストレージの使用量の増加に応じて閾値が大きくなり，Pod A と Pod B は使用量の増加が少なかったので閾値が小さくなっている．また，提案適合後に 1 つの Pod が閾値に到達した際に再び閾値を式 2 により決めなおす．

$$\mathbb{GR} = \frac{IEPV}{TI} \tag{3}$$

[*10] https://x.gd/ctITc

GR　Growth Rate(増加割合)
IEPV　Increase in Each PV(PV の増加量)
TI　Total Increase(全体の増量)

先ほど説明にあった増加割合は式 3 によって求める．

ユースケース・シナリオ

本稿では EC サイトの運営に Kubernetes と NFS サーバーを採用しているユースケースとする．

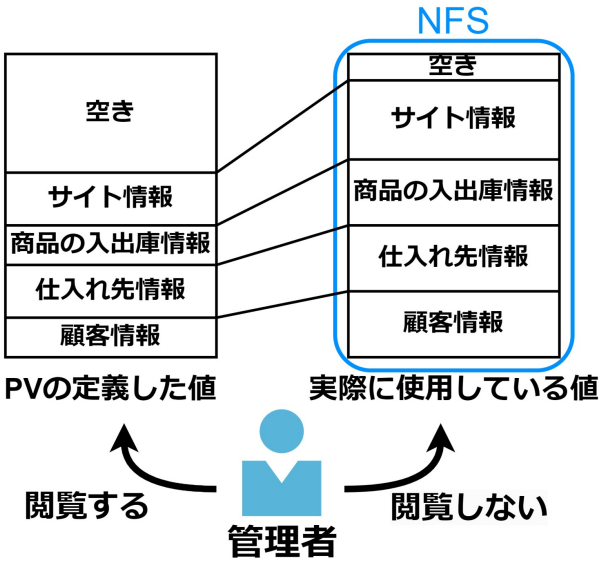

図 5　PV の定義値と実際に使用している値

図 5 は PV の定義値と実際に使用している値を示したものである．この場合，EC サイトの Pod が使用する PV の容量を超えていることに管理者が気づきにくい．その超過が原因で，NFS サーバーストレージの容量も超過してしまっていることが特定された．そこで管理者は PV が効力を持たせた．これは，PV の容量制限をより厳格に適用し，Pod が割り当てられた容量を超えた場合にデータの書き込みを制限することを意味する．しかし，効力を加えたことで NFS サーバーストレージ全体を見ると余裕があるものの，1 つの Pod の PV が指定した閾値まで達した場合にデータの書き込みができなくなってしまう．

図 6 は Pod C が閾値に達してデータの書き込みができなくなった時の各 Pod のストレージ使用状況の例である．図 6 のように閾値に到達している Pod がある一方で，Pod によっては空き容量もある．

4.　実装

実装では提案手法をもとに，新たなソフトウェアを開発言語である Python3.9 にて作成した．開発したソフトウェア（get_Poder）内のファイルがどのような機能を持っているのか説明する．

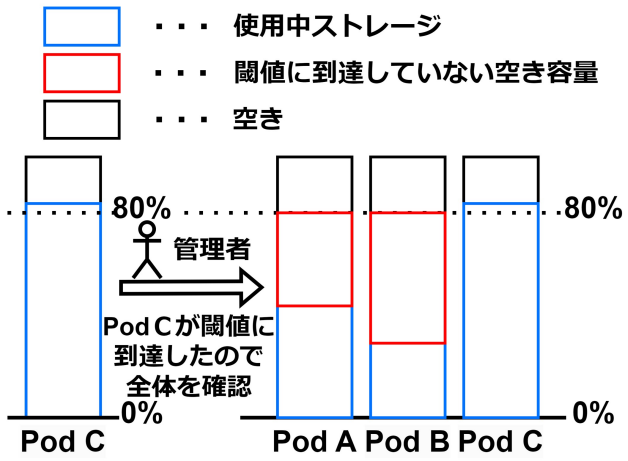

図 6　1 つの Pod の PV が閾値に到達した際の全 Pod のストレージ使用量

4.1　現在のストレージ使用状況を把握する

get_Pod_information.py では Kubernetes クラスター内のすべての Pod のストレージ使用量と Pod に紐づいている PVC(Persistent Volume Claim) のキャパシティ情報を取得し，表示している．そのような機能を実装するために行った手順は以下の通りである．

(1) ローカルの kubeconfig ファイルを読み込み，Kubernetes クラスターに接続する．
(2) Kubernetes の Core V1 API クライアントインスタンスを初期化する．
(3) クラスター内の全ての Namespace に存在する PVC のリストを取得し，各 PVC のストレージキャパシティ情報を読み取る．
(4) 全ての Namespace に存在する Pod のリストを取得する．
(5) ”du -sh”コマンドを使用して，Pod 内のストレージ使用量を取得する．
(6) ”du”コマンドからの出力を解析し，使用量の情報を取得してそれを表示する．

以上の手順で Kubernetes クラスター内のすべての Pod のストレージ使用量と Pod に紐づいている PVC のキャパシティ情報を取得し，表示しているが途中でエラー発生時の例外処理やエラーメッセージの出力を行っている．

また，get_NFS サーバー_information.py では Kubernetes クラスターにおける NFS サーバーストレージ情報を取得し，表示している．そのような機能を実装するために行った手順は以下の通りである．

(1) 関数を使用して，kubeconfig ファイルからクラスターへの接続情報を読み込む．

(2) Core V1 API のクライアントインスタンスを初期化し，API との通信を可能にする．

(3) クラスター内の全 Namespace にある Pod のリストを取得する．

(4) 各 Pod に対して”df -h”コマンドを実行して，Pod が利用するストレージの使用量を取得する．このコマンドはディスクのファイルシステムのディスクスペースの使用量を表示している．

(5) ”df”コマンドの出力から必要なストレージ情報（サイズ、使用済み容量、利用可能容量、使用率）を取得し，出力する．

以上の手順で Kubernetes クラスターにおける NFS サーバーストレージ情報を取得し，表示しているが途中でエラー発生時の例外処理やエラーメッセージの出力を行っている．

4.2 閾値を変更する

capacity_calculation.py では，Kubernetes クラスター内の各 Pod のストレージ使用量とそれに関連する PVC のキャパシティ情報を取得し，提案手法を反映させた計算をして新たな閾値を表示している．そのような機能を実行するために行った手順は以下の通りである．

(1) Kubernetes クラスターに接続し,CoreV1API を使って API を初期化する．

(2) クラスター内の全ての Namespace にある PVC のリストを取得し，それぞれの PVC の容量を収集する．

(3) クラスター内の全ての Namespace にある Pod のリストを取得し,”du”コマンドを実行してストレージ使用量を計算する．

(4) 取得したストレージ使用量を表示し，各 Pod の将来の容量需要を計算する．各 Pod に対して，式 1，式 2 を用いて算出し，次の容量割り当てを予測する．

(5) 全ての Pod の合計ストレージ使用量を計算し，NFS サーバーストレージの全体的な使用状況を表示する．

以上の手順で Kubernetes クラスター内の各 Pod のストレージ使用量とそれに関連する PVC のキャパシティ情報を取得し，提案手法を反映させた計算をして新たな閾値を表示している．

図 7 は capacity_calculation.py のフローチャート図である．図 7 にある通り，Pod の Size が PVC の 80 %を超えたときのみに各 Pod の閾値を変更していることがわかる．また，プログラムは”du”コマンドを使用して実際のストレージ使用量を確認し，より正確なデータ使用量に基づいたストレージ使用量の予測を可能にしている．

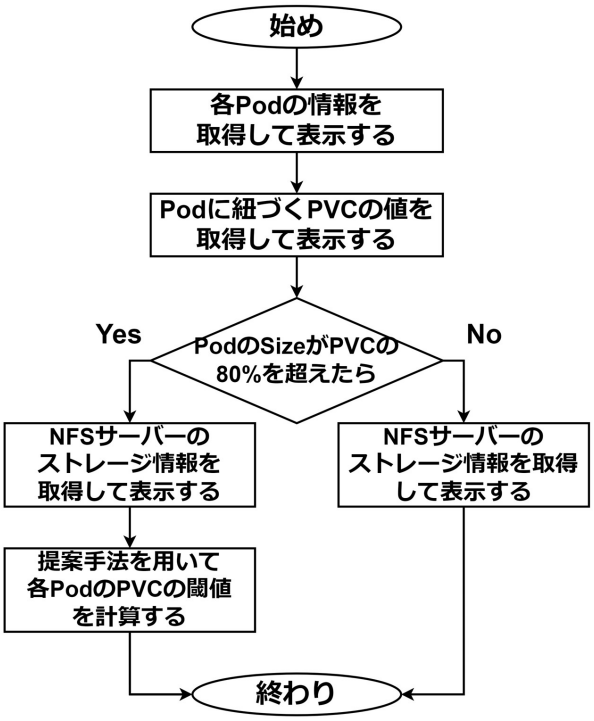

図 7 プログラムのフローチャート図

5. 評価実験

本稿の実験では EC サイトを 3 年運用することを想定として 1 つの NFS サーバーを用いて 2 つの EC サイト，ECsite A と ECsite B を構築し，各 EC サイトにて WordPress 用の Pod と MySql 用の Pod で計 4 つの Pod を用意した．その 4 つの Pod に手動でデータを追加した．追加するデータには画像，動画，テキストを用いて 0.5GB ずつ追加した．NFS サーバーで 30GB を用意し，初期値で 10GB 使用されていたので追加できるデータ量は 20GB とする．一般的に Web ページの寿命が 5 年とされている[*11]．また，EC サイトは 25GB〜100GB あれば十分とされている[*12]．よって，3 年運用するには十分なストレージ容量があるので 3 年の運用を想定し，ECsite A の WordPress の Pod で 3.92GB，MySql の Pod で 1.1GB，ECsite B の WordPress の Pod で 3.42GB，MySql の Pod で 1.1GB，4 つの Pod のトータルで 9.54GB のデータを増やすことができた．

実験環境

Ubuntu22.04 をインストールした仮想マシンを 4 台用意した．

仮想マシンをノードとして k3s を用いて作成した Kubernetes クラスターを図 8 に示す．ワーカーの仮想マシンから NFS サーバーの仮想マシンにマウントすることで図 8

[*11] https://www.sevendesign.biz/blog/afterhpcreate-repairt/
[*12] https://www.hostitsmart.com/blog/how-much-storage-do-need-for-website/

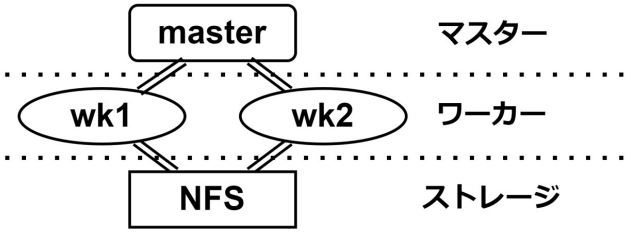

図 8 kubernetes クラスター作成時の VM の相関図

のような関係性を持ち，NFS サーバーをストレージとしての使用を可能としている．

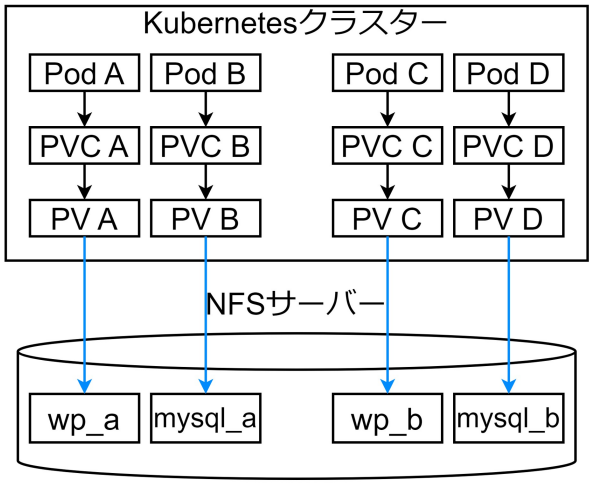

図 9 フォルダによって分割されてる NFS サーバーストレージ内

また，NFS サーバー内には図 9 のように 1 つの EC サイトごとに WordPress のデータの保存先のディレクトリと MySql のデータの保存先のディレクトリの 2 つのディレクトリを作成した．本稿の実験では EC サイトを 2 つ用意したのでディレクトリの数は 4 つになる．Pod ごとに別々のディレクトリを作成することでそれぞれのディレクトリごとにデータを保存するようになる．比較対象と提案手法を比較するために説明した実験環境を 2 つ用意した．

実験結果と分析

実際に提案手法を反映したソフトウェアを用いて実験を行った．実験を行った結果を図 10 に示す．

図 10 は評価実験を行った際，閾値 4 つの Pod のうちどれかが閾値に達した際のストレージの空き容量を示したものである．比較対象が 7.2GB，提案手法が 3.7GB となり，3.5GB 空き容量を減らし，約 50.6 %削減することができた．

6. 議論

本稿で提案した手法を用いることで，1 つの Pod の PV が閾値に達した際の他の Pod の PV の空き容量も減らすことができた．しかし，本稿の提案では将来的なデータ量の増加を予測しているので提案手法の正確性をはかることができない．過去のデータ増加量から予測することは実際の将来のデータ増加量と異なる場合に，この提案手法が機能する保証ができない．したがって，提案手法では将来のデータ量の増加に対する不確実性を引き続き考慮する必要がある．過去のデータに基づく予測はあくまでも予測であり，実際の状況に応じて対応する必要がある．

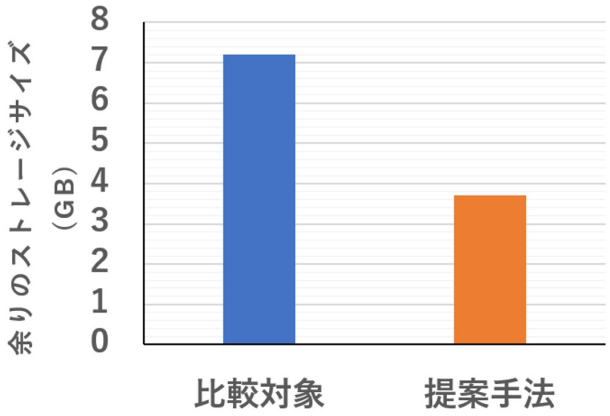

図 10 評価実験の結果

また，課題や提案の前提として，EC サイトにおける各 Pod の PV が均一であるとし比較対象にも PV が均一な EC サイトを用意した．一方で，本来の EC サイトは各 Pod が使用するデータ量を予め計算しておき，それに伴った PV を設定する．そのため，本稿での提案手法がいろんな環境においても適応されることを確認するには各 Pod に均一な PV を設定した状態ではなく，ユースケースの EC サイトに適した PV を設定し，その環境で実験を行う必要がある．

本稿の実験では提案手法を反映させた閾値の変更をする値を計算するプログラムは実装することができたが，その値を反映するプログラムは作成できなかった．よって，値はプログラムから参照して手動で変更を行った．反映するプログラムを作成することができれば，手作業がなくなるので評価対象として時間で比べることができる．

本稿の実験では，EC サイトが構築された際のデータ量と NFS サーバーの上限値は同じ値で行った．よって，実験よりも大規模な EC サイトや実験よりも小規模な EC サイトでは提案手法の正確性が高いと言えない．さらに，この提案手法の適用範囲を広げるためには，異なる規模や業種の EC サイトでの実験も重要である．例えば，異なる商品カテゴリーや顧客層を持つ EC サイトでのデータ量や PV の変動を分析し，提案手法の汎用性を確認することが必要である．

7. おわりに

EC サイトを構築する際には Kubernetes と NFS サー

バーを使用することがある．その場合，PV に効力がないのでストレージ管理が難しくなる．そこで本稿では Pod の PV のスケーリング可能な上限値を定めることで NFS サーバーストレージの超過を延期させ，サービスが停止したときのストレージの空き容量を減らす提案した．評価として，1 つ Pod の PV が閾値に達した際の NFS サーバーの空き容量を減らすことができた．実験の結果，1 つ Pod の PV が閾値に達した際の NFS サーバーの空き容量は比較対象で 7.2GB, 提案手法で 3.7GB となった．以上の結果より，提案手法のほうが比較対象と比較した際に 3.5GB 空き容量を減らし，約 50.6 %削減することができた．

謝辞 本稿の執筆に当たりご助言を賜りました, 東京工科大学コンピュータサイエンス学部先進情報専攻の平尾 真斗さん, 増田 和範さんに御礼申し上げます．

参考文献

[1] Sato, K., Soejima, T., Saito, M., Sasage, R., Kamioka, N., Goto, N. and Kawai, Y.: Development of Augmented Reality-Based Application to Search for Electric Appliances and Furniture, *2021 IEEE 3rd Global Conference on Life Sciences and Technologies (LifeTech)*, pp. 214–216 (online), DOI: 10.1109/LifeTech52111.2021.9391825 (2021).

[2] Mayrhuber, E., Krauss, O., Hanreich, M. and Stöckl, A.: Towards an Ontology and Process Mining-based System for Targeted E-Commerce Marketing Strategy Suggestions, *2023 3rd International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, pp. 1–6 (online), DOI: 10.1109/ICECCME57830.2023.10252333 (2023).

[3] Hasanah, N. A., Atikah, L. and Rochimah, S.: Functional Suitability Measurement Based on ISO/IEC 25010 for e-Commerce Website, *2020 7th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*, pp. 70–75 (online), DOI: 10.1109/ICITACEE50144.2020.9239194 (2020).

[4] Malviya, A. and Dwivedi, R. K.: A Comparative Analysis of Container Orchestration Tools in Cloud Computing, *2022 9th International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 698–703 (online), DOI: 10.23919/INDIACom54597.2022.9763171 (2022).

[5] Park, J., Choi, U., Kum, S., Moon, J. and Lee, K.: Accelerator-Aware Kubernetes Scheduler for DNN Tasks on Edge Computing Environment, *2021 IEEE/ACM Symposium on Edge Computing (SEC)*, pp. 438–440 (online), DOI: 10.1145/3453142.3491411 (2021).

[6] Islam Shamim, M. S., Ahamed Bhuiyan, F. and Rahman, A.: XI Commandments of Kubernetes Security: A Systematization of Knowledge Related to Kubernetes Security Practices, *2020 IEEE Secure Development (SecDev)*, pp. 58–64 (online), DOI: 10.1109/SecDev45635.2020.00025 (2020).

[7] Mangkhangcharoen, S., Haga, J. and Rattanatamrong, P.: Migrating Deep Learning Data and Applications among Kubernetes Edge Nodes, *2021 IEEE 23rd Int Conf on High Performance Computing Communications; 7th Int Conf on Data Science Systems; 19th Int Conf on Smart City; 7th Int Conf on Dependability in Sensor, Cloud Big Data Systems Application (HPCC/DSS/SmartCity/DependSys)*, pp. 2004–2010 (online), DOI: 10.1109/HPCC-DSS-SmartCity-DependSys53884.2021.00299 (2021).

[8] Toka, L., Dobreff, G., Fodor, B. and Sonkoly, B.: Adaptive AI-based auto-scaling for Kubernetes, *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, pp. 599–608 (online), DOI: 10.1109/CCGrid49817.2020.00-33 (2020).

[9] Toka, L., Dobreff, G., Fodor, B. and Sonkoly, B.: Machine Learning-Based Scaling Management for Kubernetes Edge Clusters, *IEEE Transactions on Network and Service Management*, Vol. 18, No. 1, pp. 958–972 (online), DOI: 10.1109/TNSM.2021.3052837 (2021).

[10] Chen, H., Tang, R., Zhao, Y., Xiong, J., Ma, J. and Sun, N.: Research on Key Technologies of Load Balancing for NFS Server with Multiple Network Paths, *2006 Fifth International Conference on Grid and Cooperative Computing Workshops*, pp. 407–411 (online), DOI: 10.1109/GCCW.2006.79 (2006).

[11] Olivares, T., Orozco-Barbosa, L., Quiles, F., Garrido, A. and Garcia, P.: The need of multicast predictive NFS servers for high-speed networks used as parallel multimedia platforms, *Proceedings International Conference on Parallel Processing Workshops*, pp. 391–396 (online), DOI: 10.1109/ICPPW.2001.951977 (2001).

[12] Soman, C. P. and Mahajan, A.: A framework for transparent relocation in distributed filesystem using R-TCP, *2008 16th IEEE International Conference on Networks*, pp. 1–6 (online), DOI: 10.1109/ICON.2008.4772610 (2008).

[13] Kai, Z.: Research on Network Data Storage Technology Based on Autonomous Controllable System, *2021 International Conference on Computer Engineering and Artificial Intelligence (ICCEAI)*, pp. 183–186 (online), DOI: 10.1109/ICCEAI52939.2021.00035 (2021).

[14] Sugiharto, G. and Asnar, Y. D. W.: Fraud Accounts Identification Modelling on Multi-Platform E-Commerce, *2021 9th International Conference on Information and Communication Technology (ICoICT)*, pp. 571–575 (online), DOI: 10.1109/ICoICT52021.2021.9527448 (2021).

[15] Konev, I., Nikiforov, I. and Ustinov, S.: Algorithm for Containers' Persistent Volumes Auto-scaling in Kubernetes, *2022 31st Conference of Open Innovations Association (FRUCT)*, pp. 89–95 (online), DOI: 10.23919/FRUCT54823.2022.9770916 (2022).

[16] Wen, T., Fan, J., Song, N. and Ji, S.: Research on key technologies for database containerized deployment, *2023 IEEE International Conference on Sensors, Electronics and Computer Engineering (ICSECE)*, pp. 1152–1156 (online), DOI: 10.1109/ICSECE58870.2023.10263524 (2023).

[17] Xiao, Z., Song, W. and Chen, Q.: Dynamic Resource Allocation Using Virtual Machines for Cloud Computing Environment, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 24, No. 6, pp. 1107–1117 (online), DOI: 10.1109/TPDS.2012.283 (2013).