Zipf法則にもとづくWordPressの対象コンテンツの限定化による移行前後の確認作業時間の短縮

西村 克己[1]　圖齋 雄治[1]　串田 高幸[1]

概要：WordPress を Docker や Kubernetes といったコンテナ環境に移行することは, クラウドや OS 環境下に関係なくシステムを継続的に運用していく上で重要である. 課題は, WordPress サイトを移行後, 確認を行うための対象が全数であると時間がかかってしまうことである. この課題を解決するために, 移行後 WordPress のページ確認対象が Zipf 分布の際に, 上位 10%のコンテンツを自動化し, 確認作業を実施した. 実験では東京工科大学 クラウド・分散システム研究室の WordPress として動作しているブログサイトを対象に行い, 評価は全数検索と抽出検索をそれぞれ 10 回実験し, 確認時間の比較を行った. 結果として, 提案ソフトウェアと全探索の作業時間を比較すると, 提案ソフトウェアでは約 2.82 秒であり, 全数検索では約 14.01 秒となり約 11.19 秒短縮することができた. また, 実装ソフトウェアを動作させるとページの存在有無について確認でき, 移行失敗の判断を行えた. この提案によって移行完了時のシステム管理者の確認における負担を軽減するものとなっている.

1. はじめに

背景

インターネット上では, Web サイトが運用されている. Web サイトは閲覧するユーザーが存在しており, ユーザーは Web サイトから情報を入手することができる [1]. Web サイトを作成するソフトウェアの一例として WordPress がある. WordPress は CMS (Contents Management System) の一種であり, Web サイトを簡単に運用することができるオープンソースソフトウェアである [2][*1].WordPress は, 記事の内容やアクセスデータの内容を, データベースである MySQL を使い, 保存する [3]. Web サイトへアクセスする際に, サイトへのアクセス数が増加してもコンテンツを閲覧し続けられるように, 管理者は維持し続けなければならない. アクセス数が増加するとページの読み込みが遅くなり, ユーザーは閲覧しようとしている Web サイトから離れてしまうため悪影響を与える [4]. Web サイトへのアクセス数増加による対策として, Web サイトをコンテナ化したものを Kubernetes で管理する方法がある.

Kubernetes とは, コンテナ化したアプリケーションを動作させるためのコンテナオーケストレーションツールである [5,6]. Kubernetes を利用することによって, アクセス数が増加したとしても負荷を分散することができるため, 障害に強い構成を作ることができる [7][*2].

コンテナ技術が発展していくことでクラウドへのシステムを移行が増加している [8]. WordPress を移行する際に, 移行中のデータ書き込みによってデータに差異が生まれないように Web サイトをダウンさせる. この時間帯はユーザーがサイトを見ることができなくなる. よって, Web サイトがダウンしている時間帯をなるべく短くする必要がある.

課題

課題は WordPress のコンテンツが保存されているデータをすべて探索し, ページの存在を確認するためには時間が増加してしまうことである. 図 1 は, サーバーの管理者が実際に全数ページを確認している様子である. 移行完了後 Web ページの確認を行うために, 図 1 のように Web ページの公開しているページ数が増えるほど, 確認項目が増える. よって移行後の確認作業の時間がかかってしまう.

仮想マシンからコンテナ環境に WordPress サイトを移行し終わった際, 移行できたかどうかの確認を行うために取られる方法として, 全数検索と抽出検索がある[*3].

表 1 に, 全数検索と抽出検索でのメリット・デメリットをあげる.

ページのコンテンツが増えるほど時間と工数がかかって

[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://ja.wordpress.org/

[*2] https://zenn.dev/esaka/articles/2d117655af1f03cf2444

[*3] https://www.stat.go.jp/teacher/survey.html

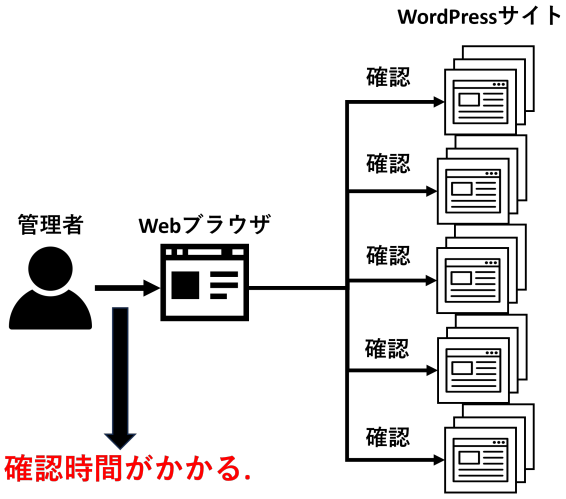

図 1 WordPress サイトを管理者が確認する様子

表 1 全数検索と抽出検索でのそれぞれのメリット・デメリット

検索種類	メリット	デメリット
全数検索	抽出検索より正確性がある	抽出検索より時間がかかる
抽出検索	時間が全数よりかからない	正確性が全数検索よりない

しまう. 実際に対象物としてあげている東京工科大学 クラウド・分散システム研究室 (以後本研究室) の WordPress のブログサイトには 2023 年 11 月 7 日現在, 固定ページで 14 件, 投稿が 323 件ある[*4].

WordPress サイトの投稿された記事一覧が保存されているデータベースを取り出すだけでは, 存在しないページがデータベース内に保存されていることや重複ページがある. そのため, どのコンテンツを確認対象として特定のデータベース内のデータを確認しなければならないのかわからない点がある.

実際に WordPress の投稿された記事一覧が保存されているデータベースのテーブルの中身を表 2 にて示す.

表 2 WordPress の記事一覧が保存されているテーブル

ID	post_date	post_name	post_status
3	2019/9/12	privacy-policy	draft
117	2020/1/7	tech-report	publish
131	2020/1/17	member	publish
150	2023/3/2	150	publish
152	2020/1/18	152-revision-v1	inherit

表 2 の項目で post_status があるが, 記事を投稿しようとして途中になっている状態を示す draft や自動保存された記事の状態, 添付されているメディアファイルに付与されるもの状態である inherit のように実際の Web サイトには投稿されていない記事もデータベースのテーブルに保存されていることがわかる.

[*4] https://ja.tak-cslab.org/

各章の概要

本稿では以下の内容で構成している. 第 1 章では, 本稿の背景と課題について述べる. 第 2 章では, 本稿の関連研究について述べる. 第 3 章では, 本稿の課題について解決するための提案方式について述べる. 第 4 章では, 提案した手法の実装について述べる. 第 5 章では, 評価実験として実験内容と実験結果と分析について述べる. 第 6 章では, 提案手法についての議論を述べる. 最後に, 第 7 章にて結論を述べる.

2. 関連研究

仮想マシンの移行においてネットワークの可用性テストを行った研究がある [9]. この研究では移行の成功率を定量化するために, ネットワーク可用性テスト及びレポートのフレームワークを設計, 実装されている. 数学的仮説を建てると移行の 87%が成功していた. この研究は, 仮想マシンでの移行の信頼性には述べているが, 仮想マシンからコンテナ環境のように動作するシステムの状況が変わった際の信頼性までは述べていない.

仮想マシンの動的マイグレーションにおいて, データマイグレーション量を減らし, ネットワークの負荷を軽減した研究がある [10]. この研究では確率予測アルゴリズムとメモリ圧縮アルゴリズムを提案し, 2 つのアルゴリズムのデータ構造と設計思想を述べていた. これは合計の移行時間とダウンタイム時間が短縮され, 移行後のパフォーマンスが向上することは証明されているが, 移行後の確認の話までは述べていない.

一貫性の優先度について述べている研究がある [11]. この研究ではすべてのレプリカではなく, 一貫性の優先度の高いレプリカのみが更新し, 更新の優先順位の提案をしている. ただし, これはモバイルオーバーレイネットワークの話をしており, WordPress のようなアプリケーションの話までは述べていない.

3. 提案

提案方式

本提案では, 全数を確認するのではなく, 優先度の高いコンテンツを見に行くこととする. そのようにすることで, 確認の時間を減らすことができる. 優先度の目安として今回は WordPress で公開しているページ総数のうち, 上位 10%のアクセス数のページを確認する. 実際に提案方式を示したものとして図 2 がある. WordPress で公開しているページ総数の中から, アクセス数上位 10%のコンテンツを確認対象として置く. 確認対象が Zipf 分布の際に, アクセス数上位 10%のコンテンツのページを対象として確認することで, コンテンツの一貫性を保つことができる [11]. これにより, ブログの一貫性を保ちながら, 全数検索のような正確性で移行の確認を行うことができる.

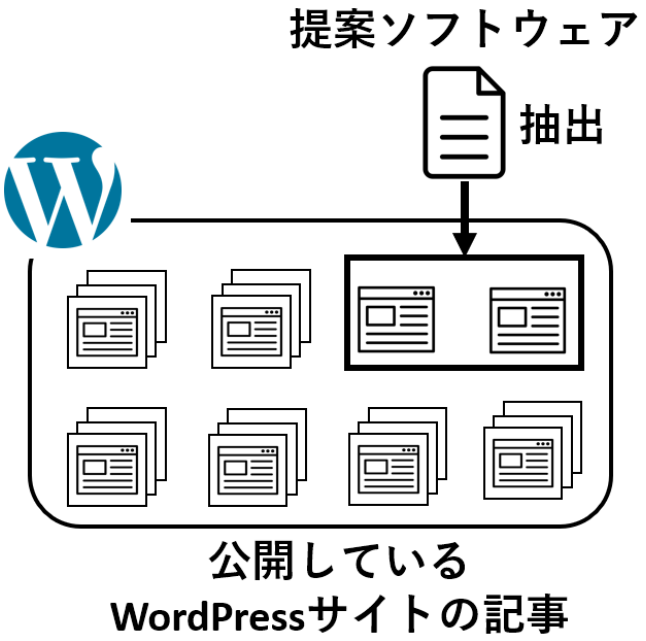

図 2　提案方式

また本提案で扱っている Zipf 分布が本研究室のブログサイトにおいて, 公開しているページ総数である 337 件のうち上位 10%のアクセス数の分布に則っているか回帰分析で確認した. 結果を図 3 に示す.

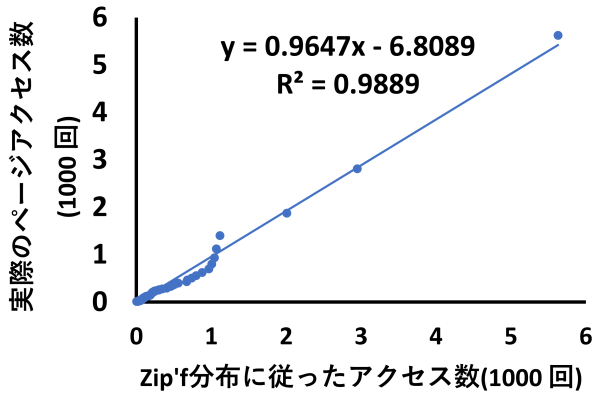

図 3　回帰分析を行ったデータ

縦軸は実際の公開されているページのアクセス数を示したものとなっている. 横軸は 1 番アクセス数が多かったページから zipf 分布に従ったアクセス数を示したものとなっている. 重解式の当てはまりの良さを示す指標である決定係数の R^2 が 0.9889 となっている. 1 に近いほど分析の精度が高いため, 限りなく分布に近似していることがわかる. このことから, Zipf 分布は今回の提案で使用できる.

ユースケース・シナリオ

本稿では, 本研究室のブログサイトを対象物としている[*4]. 仮想マシンにインストールされた WordPress で動作しているブログサイトをコンテナ化することを事例においている. 移行前の仮想マシンで Web サイトを運用している状態を図 4 に示す.

図 4　移行前の状態

図 4 のように, 仮想マシン上で WordPress と MySQL が共存した状態で動作させていた際, 背景でも述べた通りアクセス数が増加したり, システム障害によって動作を停止してしまう場合がある [*5].

このような事例を防ぐためにも, 図 5 のように Kubernetes 上に WordPress と MySQL を移行し, スケーリングを自動的に行うことで, Web サイトを効率よく運用することができる.

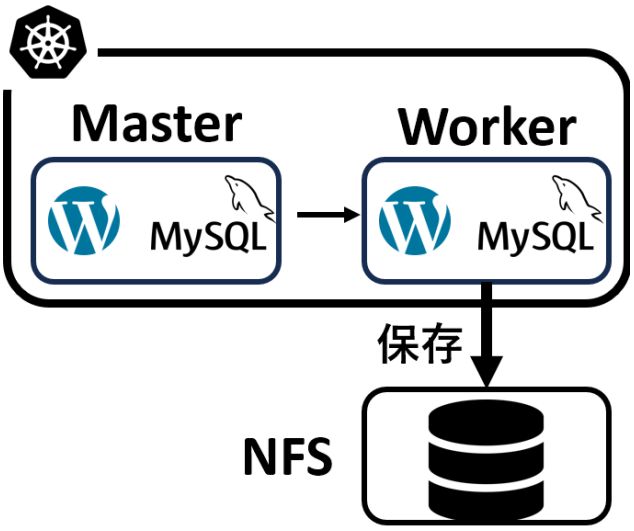

図 5　移行後の状態

また, WordPress はオープンソースソフトウェアであり,

[*5] https://www.asahi.com/articles/ASP5B7394P5BUNHB009.html

プラグインを導入することができる．本研究室では統計情報を取得するために WP Statistics というプラグインを導入している[*6]．このプラグインはアクセス数を取得することができ, その内容はデータベース内のテーブルに保存される．保存されたデータを元に公開しているブログサイトのページの上位 10%分のアクセス数を抽出して, ページの確認を行うこととする．

4. 実装

実装では, 移行後の Kubernetes クラスター上 (k3s) にある MySQL の pod を, 開発言語である Python のライブラリとして公開されている MySQL Connector を用いて探索した[*7]．探索対象は公開されている WordPress のページ, アクセス数の取得である．この情報を元に, 公開されている WordPress のページのうち, アクセス数の上位 10%を取得し, システムの移行後のページの存在の確認を行っている．実際の流れを図 6 にて示す．

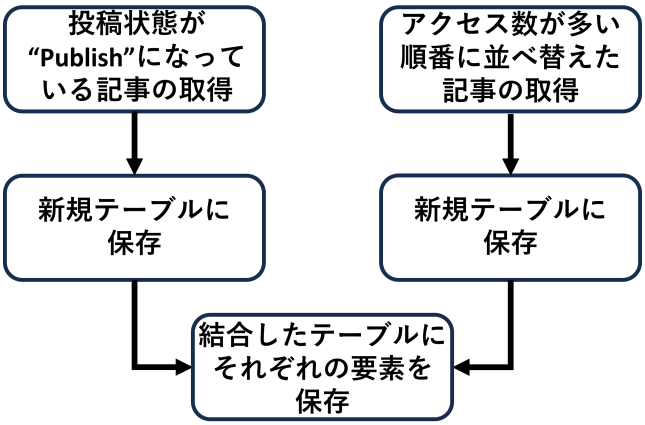

図 6 ソフトウェアの概要

実装ソフトウェアは, 移行後の Kubernetes クラスター内でデータベースとして動作している MySQL の情報を読み取り, 動作するものとなっている．以下にて各動作について説明を行う．

4.1 投稿状態の判定

課題の表 1 にて示したが, wp_post はステータス (公開, 非公開, 下書き, 自動保存) に関係なく, すべての記事が同じテーブルに保存されてしまう．そこで今回はステータスが公開状態であることを示す Public になっている p ページのみを検索対象として抜き出して, 記事の取得を行うこととした．その後取得した結果を新しいテーブルとして保存することとした．

[*6] https://wordpress.org/plugins/wp-statistics/
[*7] https://dev.mysql.com/doc/connector-python/en/

4.2 アクセス数の判定

本研究室のブログサイトには統計情報を取得するために, WP Statistics というプラグインを導入している．このプラグインにて取得したアクセス数は wp_statistics_pages というテーブルに保存されている．実際に保存されている一部を表 3 に示す．

表 3 wp_statistics_pages のテーブルに保存されている内容の一部

page_id	uri	type	date	count	id
27535		loginpage	2023-01-17	172	0
25718	/feed	feed	2022-12-17	129	0
25887	/feed	feed	2022-12-21	119	0
36193	/archives/460	post	2023-04-23	63	460
36194	/archives/460/	post	2023-04-23	63	460

表 3 の uri が空欄になっている項目と/feed の項目は, 実際には存在しないページであるにも関わらず, アクセスされた情報として記録されている．また, 表の下から 1 番目や 2 番目の項目にて, uri が同じであるものや uri 最後の”/”有無であるように page_id が異なるがために別ページとしてカウントされている．アクセス数順に並べ替えを行っても正確に移行対象としてのページの抜き取ることができない．そのために, ソースコード 1 のように処理を加えた．

ソースコード 1 アクセス数が別カウントとして保存されたページの修正

```
1    INNER JOIN wp_nissy_posts posts ON
2        SUBSTRING_INDEX(SUBSTRING_INDEX(counts.
             cleaned_uri, '/', -1), '(', 1) =
             SUBSTRING_INDEX(SUBSTRING_INDEX(
             posts.guid, '=', -1), '(', 1)
3        OR
4        SUBSTRING_INDEX(SUBSTRING_INDEX(counts.
             cleaned_uri, '/', -1), '(', 1) =
             SUBSTRING_INDEX(SUBSTRING_INDEX(
             posts.guid, '=', -1), ')', 1);
```

4.1 にて説明した投稿状態の判定の結果と今回行ったアクセス数の判定の uri の中身が同じである場合は同一ページとしてカウントし, 足し合わせたものをアクセス数として計算して, 新規テーブルに保存することとした．

4.3 結合テーブルの作成

4.1, 4.2 にて投稿状態, アクセス数の判定を行い, 新規テーブルに保存した．投稿状態が公開になっているものでかつ, アクセス数順に並べ替えた要素を結合し, 最終的に新規テーブルに保存することとした．結合後に作成したテー

表 4　4.1, 4.2 の内容を結合したテーブルの一部

id	url	total_count	title	post_status
1	/archives/460	5094	タイトル A	publish
2	/tech-report	2745	タイトル B	publish
3	/archives/901	1884	タイトル C	publish
4	/about/member	1101	タイトル D	publish
5	/about	1021	タイトル E	publish

ブルの内容の一部を表 4 に示す.

表 3 のような, 同一ページが別カウントとして処理されず, 表 4 のように, ページごとのアクセス数順に並べ替えられており, ここから公開されているページの上位 10%のアクセス数を持つ要素を探索することとした.

5. 評価実験

実験では実装したソフトウェアを動作させ, 移行後のページが存在しているかどうかの確認を行う. ここでの確認とは, linux の curl コマンドを使用し, Web ページのステータスコードで 200 番台, 300 番台の返答があればページが存在しているものとして定義し, ステータスコードが 400 番台, 500 番台のページの際にはページが存在しないものとして定義する [12]. この定義付けを提案ソフトウェアで判断して今回対象物としてあげている本研究室の WordPress サイトが移行できているかどうかの確認を行うこととする. 今回課題として, 移行後の確認を行う際に全数検索では時間がかかることをあげているため, 実験では全数検索と抽出検索との時間の比較を行うこととした. 作成した実装ソフトウェアは Python で実装しており, Python のライブラリに time モジュールがある[*8]. time モジュールはプログラムの開始時間と終了時間を計測することができる. この時間を計測し, 実際に全数検索と抽出検索での比較を行った.

また移行するページの確認を行う際に, 移行前にあったページが移行後に欠損していると問題である. そのため, 移行前にあったページが移行後の確認の際ページ移行できていないことを確認できるかの検証を行うこととした. 検証方法として, 学生 1 人に対して移行確認対象である上位 10%のページの中から, 実装ソフトウェアを実行する者に伝達せず, 意図的に移行後の Kubernetes クラスター上で動作している WordPress サイトのとある 1 ページの公開状況を非公開に設定してもらった. 実装したソフトウェアが正しくステータスコード 400 番台・500 番台のページを発見し, 移行の失敗を検知できるかの実験を行った.

[*8] https://docs.python.org/ja/3/library/time.html

実験環境

実験環境を以下に示す. 移行元の仮想マシン, コンテナ環境である Kubernetes クラスター, Kubernetes で動作させる WordPress サイトのコンテンツを保存する NFS サーバーの 3 つの環境となっている.

本研究室の WordPress サイトは一日ごとにフルバックアップでコンテンツデータを保存している. このバックアップデータを移行元の仮想マシン, 移行後として動作させる Kubernetes クラスターのデータ保存場所として用意した NFS サーバーにそれぞれコンテンツデータをコピーし, 移行後実際にアクセス数を元にページ移行後の確認を行うこととした.

以下に, それぞれの仮想マシンの構成要素を示す.

- 移行元の仮想マシン構成要素
  OS: Ubuntu Server 22.04
  vCPU: 1Core
  RAM: 1GB
  HDD: 30GB
- Kubernetes クラスタのマスター, ワーカーノード構成要素
  OS: Ubuntu Server 22.04
  vCPU: 2Core
  RAM: 2GB
  HDD: 30GB
- NFS サーバ構成要素
  OS: Ubuntu Server 22.04
  vCPU: 1Core
  RAM: 1GB
  HDD: 32GB

実験結果と分析

実際に提案ソフトウェアにて全数検索, 抽出検索を行ったものを 10 回実験した. 確認の動作を行った平均時間を図 7 に示す.

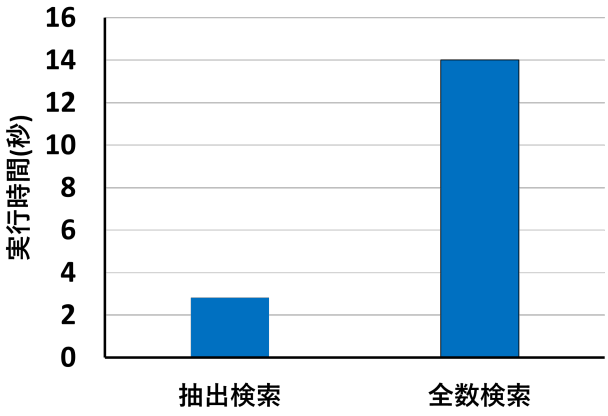

図 7　全数検索との比較

提案ソフトウェアは平均約 2.82 秒, 全件検索は平均約 14.01 秒だった. 全検索と比較して, 提案ソフトウェアでは検索時間が 11.19 秒短縮し, 約 5 分の 1 になった. 検索時間を削減することにより, 移行時の確認にかかる時間を減らし, 移行確認時のチェック簡易化に貢献する.

また前述の通り, 学生にお願いした意図的に移行後のページの 1 ページを非公開にし, 実際に確認作業を実装ソフトウェアにて行った. 実装ソフトウェアの出力結果をソースコード 2 に示す.

ソースコード 2 実装ソフトウェアの出力結果

```
OK Pages:
1: /archives/460
2: /tech-report
3: /archives/901
4: /archives/896
5: /about
6: /archives/1064
7: /about/member
8: /archives/498
9: /archives/1125
10: /archives/3064
11: /archives/2898
12: /archives/3187
13: /archives/2252
14: /dojo
15: /archives/2069
16: /publish-assets
17: /about/about-cdsl
18: /archives/848
19: /archives/527
20: /join-us
21: /about/it-facility
23: /archives/2464
24: /archives/1342
25: /archives/786
26: /archives/1049
27: /about/facility
28: /archives/3027
29: /archives/1211
30: /archives/1180
31: /about/tools
32: /archives/2230

NG Pages:
22: /archives/770
```

ソースコード 2 にあるように, ページの存在有無について確認を行った結果, /archives/770 というページがステータスコード 404 になっており, NG Pages として表記があった. 学生に聞いたところ該当ページの公開状態を非公開にしたとのことであり, 実際にページの存在有無について確認を行うことができた.

6. 議論

本稿の提案方式では, ページが存在しているかどうかまでの確認は行っているが, ページの中身の一致までの確認を行っていない. Web サイトは html で構成されている. 今回使用した技術として curl コマンドがある[*9]. curl コマンドでは, Web サイトの中身について確認することができる. そのため curl コマンドで公開されているページのアクセス数上位 10%のコンテンツの中身を移行前, 移行後で比較することによってより信頼度の増す提案になる.

今回は公開されているページのアクセス数上位 10%のコンテンツを取得してきた. 図 7 にあるように, 検索時間は全数検索の 10 分の 1 になるはずだが, 実際にはその倍の 5 分の 1 の短縮になった. この件に関して, 2 つの原因が挙げられる. 1 つ目に curl コマンドでのキャッシュである. 今回実装したソフトウェアは, ページの有無の確認を行うためにデータベース内に保存されているデータから, curl コマンドで確認を行っている. この動作を検索時間の平均を取るために 10 回プログラムを実行した. その際, 検索結果のキャッシュが一時的に保存される. 2 回目以降実装ソフトウェアを実行した際, 1 回目に検索した際のキャッシュが利用されるため, 検索時間が速くなるが, 今回は予測以上に時間の差が見られなかった. プログラムを実行する際, 1 回目の検索結果のキャッシュが 2 回目以降の検索には影響を与えなかったため, より大規模なサイトでこの提案を検証すると時間の差が見られる. 2 つ目にステータスコードである. 今回実装したソフトウェアでは curl コマンドでのステータスコードを元に判断を行っている. curl コマンドではステータスコードを取得する際, HTTP ヘッダーのみ取得し, ページの本文をダウンロードしない. サーバーの応答が遅い場合, 検索時間が予測よりも長くなる. この場合, ページ本文の内容を含めて比較するとより違いが見られる.

移行後の確認の対象として公開されている上位 10%のコンテンツを取得してきたが, 工場での抽出検索であるように, 無作為で取得するやり方も一つの方法としてあげられる[*10]. 無作為で検索対象を抽出しつつ, 正確性と検索時間を担保できる 1 つの方法である.

今回の提案では, 公開されている WordPress ページのアクセス数上位 10%を検索対象としておいている. 今回の検索対象である本研究室のブログサイトは 2019 年 9 月 23 日に公開がされており, 検索期間は過去 4 年間の累積のページアクセス数を元に提案を適用した. ただし, 過去アクセスがされたページがあったとして, 今現在はアクセスがされていないページがあることが考えられる. 現在アクセスがされていないページが, 今後アクセスされるかどうかは明らかではない. 例として, アクセス数取得の期間を 1 ヶ月と定め, 表したものを表 5 に示す. 表 5 の url は/a/b/c/d/e の記事, before_count は, 1 ヶ月アクセス数取得前の数, after_count は, 1 ヶ月アクセス数取得後の数, diff_count は, 500,400,300,200,100 のように, 1 ヶ月アクセ

[*9] https://curl.se/
[*10] https://www.ouj.ac.jp/mijika/tokei/xml/k3_02026.xml

表 5　アクセス数の取得期間からの差分の例

url	before_count	after_count	diff_count
/a	2603	3205	500
/b	4790	5257	400
/c	831	1136	300
/d	781	970	200
/e	2603	1124	100

ス数取得後の数から 1 ヶ月アクセス数取得前の数を引いた数となっている. diff_count のように, WordPress で投稿したそれぞれの記事の 1 ヶ月間にあったアクセス数のデータからアクセス数上位 10%の対象となるページを検索対象としておくと, サイトの傾向を考慮できる.

7. おわりに

アプリケーション移行後の確認には時間がかかる. その負担を減らすために, 今回はアプリケーションの中でも WordPress に対象を絞った. 移行後の確認の項目数を削減することにより, 移行完了時の確認を簡易化することとした. 実際に検索対象に該当するページにおいてページの存在有無を実装ソフトウェアにて確認することができた. この提案手法を基に公開しているページ数の全数検索と抽出検索を比較した. 全検索では平均約 14.01 秒であったのに対し, 抽出検索では平均約 2.82 秒であり, 検索時間を約 5 分の 1 に短縮できた. 学生 1 人に移行対象のページに対して公開状況を非公開に設定してもらい, 実際に実装ソフトウェアを動作させたところ, ページの存在有無について確認を行うことができ, 移行失敗の判断を行えた.

謝辞

本稿の執筆に当たりご助言を賜りました, 東京工科大学大学院バイオ・情報メディア研究科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] Nakajima, A. and Aoki, Y.: Transformation of web event sequences for analysis of users' Web operation, *IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.99CH37028)*, Vol. 4, pp. 111–116 vol.4 (online), DOI: 10.1109/ICSMC.1999.812385 (1999).

[2] Ruohonen, J.: A Demand-Side Viewpoint to Software Vulnerabilities in WordPress Plugins, *Proceedings of the 23rd International Conference on Evaluation and Assessment in Software Engineering*, EASE '19, New York, NY, USA, Association for Computing Machinery, p. 222–228 (online), DOI: 10.1145/3319008.3319029 (2019).

[3] Kumar, A., Kumar, A., Hashmi, H. and Khan, S. A.: WordPress: A Multi-Functional Content Management System, *2021 10th International Conference on System Modeling Advancement in Research Trends (SMART)*, pp. 158–161 (online), DOI: 10.1109/SMART52563.2021.9675311 (2021).

[4] Grassi, G., Teixeira, R., Barakat, C. and Crovella, M.: Leveraging Website Popularity Differences to Identify Performance Anomalies, *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, pp. 1–10 (online), DOI: 10.1109/INFOCOM42981.2021.9488832 (2021).

[5] Hardikar, S., Ahirwar, P. and Rajan, S.: Containerization: Cloud Computing based Inspiration Technology for Adoption through Docker and Kubernetes, *2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC)*, pp. 1996–2003 (online), DOI: 10.1109/ICESC51422.2021.9532917 (2021).

[6] Carrión, C.: Kubernetes scheduling: Taxonomy, ongoing issues and challenges, *ACM Computing Surveys*, Vol. 55, No. 7, pp. 1–37 (2022).

[7] Yunyun, Q., Chen, P. and Tan, D.: Research on Elastic Cloud Resource Management Strategies based on Kubernetes, *2022 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA)*, pp. 441–448 (online), DOI: 10.1109/AEECA55500.2022.9918897 (2022).

[8] Benjaponpitak, T., Karakate, M. and Sripanidkulchai, K.: Enabling Live Migration of Containerized Applications Across Clouds, *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pp. 2529–2538 (online), DOI: 10.1109/INFOCOM41043.2020.9155403 (2020).

[9] Kukrál, T., Kozák, M., Hégr, T. and Boháč, L.: VM migration measurement and failure detection, *2015 38th International Conference on Telecommunications and Signal Processing (TSP)*, pp. 285–288 (online), DOI: 10.1109/TSP.2015.7296269 (2015).

[10] Mingsong, S. and Wenwen, R.: Improvement on dynamic migration technology of virtual machine based on Xen, *Ifost*, Vol. 2, pp. 124–127 (online), DOI: 10.1109/IFOST.2013.6616868 (2013).

[11] Su, Z., Katto, J. and Yasuda, Y.: Priority Based Selection to Improve Contents Consistency for Mobile Overlay Network, *2009 IEEE Wireless Communications and Networking Conference*, pp. 1–5 (online), DOI: 10.1109/WCNC.2009.4917921 (2009).

[12] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and Berners-Lee, T.: Hypertext transfer protocol–HTTP/1.1, Technical report (1999).