Vol.2018-SE-200 No.8
2018/12/3

LAMP を対象にした運用中障害発生事例の調査

名倉 正剛[1,a)]　高田 眞吾[2]

概要：Web アプリケーション等のサーバサイドアプリケーションでは，運用中の負荷の変動を運用開始前に予測することが困難である．このため，運用開始前に十分にテストが実施されていても，運用開始後にパフォーマンスに対する障害をはじめとして，各種の障害が発生する場合がある．サーバサイドアプリケーションの実行は，サーバミドルウェアや OS などのいろいろな構成要素の影響を受ける．特にオープンソースソフトウェアを組み合わせて構築されている場合には，障害発生時に調査すべき対象が多岐にわたり，原因特定が困難である場合が多い．本研究では原因特定の支援を目指して，サーバサイドアプリケーション運用中に発生した障害発生に対する原因特定や対策の手順を，Q&A サイトから LAMP に対する事例を中心に調査した．具体的には LAMP に対する質問（約 2500 件）から目視によりソフトウェア障害への対応に関連する質問を抽出し，原因により分類を行った．そして障害状況を実際に再現して確認した結果，LAMP を構成するそれぞれのソフトウェア構成要素に対して，障害発生の原因とそれにより現れる障害状況がある程度限定されることがわかった．本報告ではこの結果について述べたのち，この結果を利用した障害原因推定手法の概略について述べる．

キーワード：サーバサイドアプリケーション，LAMP，障害原因分析，Q&A サイト

1. はじめに

サーバサイドアプリケーションの運用中に障害が発生した場合，障害原因の特定が困難になる場合がある．

障害原因がハードウェアの場合は，ベンダーがハードウェアとともに提供している診断プログラム [1] や，ベンダにより別途提案されている原因分析のための手法 [2] [3] が実装された管理ソフトウェア [4] を利用することで，原因特定を容易に実施できる [5] ことが多い．

しかし，ソフトウェアに発生した障害が原因の場合，原因特定が困難になる．Web サーバなどのサーバサイドアプリケーションの実行基盤は，オープンソースソフトウェアを組み合わせて構築することが一般的である．ベンダが存在するハードウェアと異なり，監視ソフトウェアや原因分析のための手法が提供されているとは限らない．このため障害発生時には，構成要素のログファイルや関連する設定ファイルなどを調査することになる．この際に，構成要素によって調査対象が異なり，しかも多岐にわたる．従って，一般的にソフトウェア障害に起因する運用障害は，原因特定に時間を要する．

さらにサーバサイドアプリケーションでは，運用中の状況変化が予期せぬ挙動を引き起こし，障害が発生することも多い [*1]．特に Web アプリケーションは，インターネットに接続されたクライアントであれば誰でもアクセスできるため負荷変動を予測できず，負荷変動に起因する障害発生は回避が困難であると言われている [8]．

本研究では，運用中に発生した障害に対する原因特定の支援を目指して，Web アプリケーションのパフォーマンス障害に対してどのような手順で原因特定や対策を実施しているのかを調査した．具体的には，インターネット上の Q&A サイト（Stack Overflow/Server Fault/Super User）から LAMP に対するパフォーマンス障害発生事例を中心に調査を実施した．LAMP は Web アプリケーションを動作させる際に利用するオープンソースソフトウェア群（OS，Web サーバ，データベース，アプリケーションを動作させるプログラム実行系）の総称であり，狭義にはそれぞれ Linux，Apache，MySQL，PHP を指す．まず，2018 年 4 月時点で収集可能な Q&A サイトの全質問から，LAMP に対する質問に関連するスレッド（約 2,500 件）を

[1] 南山大学理工学部
[2] 慶應義塾大学理工学部
[a)] nagura@nanzan-u.ac.jp

[*1] 例えば IPA/SEC による 2017 年の日本国内情報システム障害状況の報告データ [6][7] から障害原因に関する記述を確認し分類すると，全 67 件の障害からサーバサイドアプリケーションが関係していないと判断できる 3 件を除いた 64 件中，ハードウェア故障が 7 件，アプリケーションバグが 11 件，パフォーマンス障害が 6 件，設定変更や作業に起因する障害が 27 件，原因不明なものが 13 件であった．

抽出した．そして，目視によりソフトウェアの運用時の障害に関連するスレッドを抽出し，原因により分類した．次に，分類したスレッドに記載された障害状況を実際に再現し，障害状況を確認した．その結果，LAMP を構成するそれぞれの構成要素に対して，障害発生原因とそれにより現れる障害状況がある程度限定されることがわかった．本論文では，この調査結果について報告する．そしてこの結果に基づき我々が考える障害原因推定手法の概略を述べる．

2. 関連研究

サーバサイドアプリケーションは一般的なデスクトップアプリケーションより設定変更などによる影響が障害を引き起こしやすいことが指摘されている [9]．そのような環境で発生した障害の原因診断は，一般的にログコンテンツを解析することで実施する [10]．しかし，障害状況を示すログ出力が複数のログファイルに広範に広がっていたり，監視のためのソフトウェアを利用していてもソフトウェア上にたくさんのアラームが生じていたりすることで，その解析だけでも非常に手間を要することが報告されている [11][12]．このように，サーバサイドアプリケーションに発生した障害の原因特定や対策実施は容易ではない．

そこで Sayagh らは，LAMP 環境を構成するソフトウェア群の設定ファイル間で，障害に対して依存関係のあるオプションを推定する方法を提案している [13] ．この方法では，LAMP 環境を構成するソフトウェア要素に対してコードスライスを実行し，スライス依存グラフを事前に作成することで，障害発生時にどのソフトウェア要素のオプションに依存して障害が発生したのかを特定する．

3. LAMP ソフトウェアスタックと障害の伝搬

3.1 LAMP ソフトウェアスタック

今日，サーバサイドアプリケーションの形態として，Web アプリケーションが最も普及している．LAMP は，Web アプリケーションを実現するために，一般的に利用されるソフトウェアスタックの一つである [14]．**図 1** に LAMP のソフトウェアスタックを示す．このソフトウェアスタックは Linux（L : OS），Apache（A : Web サーバ），MySQL（M : データベース），PHP（P : 実行エンジン）の各レイヤから構成され，この上で Wordpress や Drupal 等のアプリケーションを動作させる．LAMP はこのような Web アプリケーションを動作させるための基盤として利用されるソフトウェア群の総称であり，広義には個別のソフトウェア要素について同種の別ソフトウェアを含む（例えば，図 1 のように，MySQL の代わりに PostgreSQL を利用したり，Linux の代わりに Windows を利用したりするなど）．

3.2 スタック間の障害の伝搬

LAMP に代表されるサーバサイドアプリケーションのためのソフトウェアスタックでは，個々のソフトウェア構成要素が層状に構成され，上位の構成要素の動作が下位の構成要素に依存する．その結果として，スタック全体の振る舞いや，スタック上で動作する Web アプリケーションの振る舞いは，個々の構成要素の組み合わせに複合的に依存する．それぞれの構成要素はいろいろな種類のアプリケーションでの利用を念頭に，共通的なプラットフォームとして提供されているため，設定オプションによりその振る舞いをカスタマイズできる．このため，誤った設定オプションの指定により，設定エラーを引き起こすことが容易に起こりうる．誤った設定は誤作動やクラッシュの原因となるが，運用開始までにそのようなエラーが顕在化しないこともあり，運用開始後にエラーが発生することも多い．

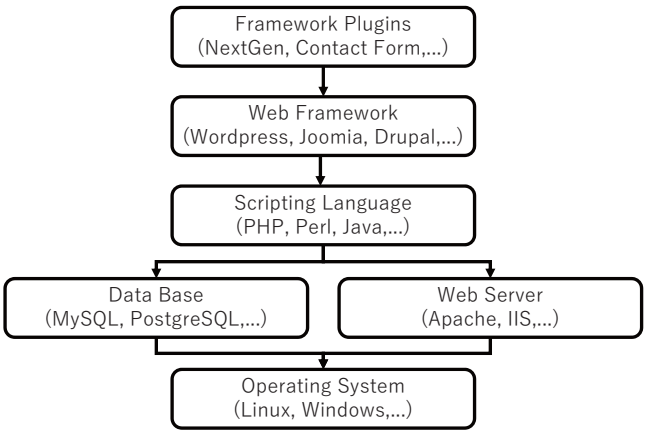

図 1 LAMP スタック（文献 [13] Figure 1 より抜粋）

スタックのあるレイヤで設定エラーが発生した場合は，そのレイヤを構成するソフトウェアのログから障害原因を解析する．障害の状況が現れた構成要素と障害原因の設定エラーを含む構成要素が同一レイヤである場合（SLCE : Single Layer Configuration Error）だけではなく，それらが異なったレイヤに存在する場合（CsCE : Cross-stack Configuration Error）もある．

CsCE は，スタック内のあるレイヤの構成要素を原因とした設定エラーの影響が，他のレイヤに波及することである．原因となるレイヤにエラーが発生していない場合でも，そのレイヤの振る舞いが他のレイヤに影響し，結果的に別レイヤにエラーを引き起こすことがある．その結果として前述の Sayagh らは，CsCE の場合にシステム停止（クラッシュ）に至る割合が，SLCE に比べて高いと報告している [13]．特にパフォーマンスに関する設定によっては，例えばあるレイヤでの接続許容量を増やすことによって，上位レイヤの処理に必要なリソースが増え，上位レイヤにエラーを引き起こすようなこともある．このように，LAMP を始めとするソフトウェアスタックで発生する障害が CsCE の場合は，障害発生と原因の因果関係の分析が難しくなる．さらに，レイヤごとに運用を行う管理者が異なることあり，そのことも CsCE の解決を困難にする．

4. 事例調査・結果

4.1 概要

サーバサイドアプリケーションに発生した原因特定の支援を目指して，実際に Web アプリケーションの運用中に発生したパフォーマンスに関する障害事例より，どのような障害に対してどのような手順で原因特定や対策を実施しているのかを調査した．

調査は，Q&A サイトのユーザコミュニティでの質問と回答が記載されたスレッドからソフトウェア構成に起因する障害を調査し，目視によりレイヤごとに障害状況をカテゴリ分けした．そして，次にカテゴリ分けした障害状況を LAMP 環境を用意して実際に再現した．Q&A サイトのスレッドにはログやコマンド実行結果を利用した原因判別のための解析手順が記載されていた．そこで再現した障害に対して，実際にその手順を実施することにより障害原因を特定できることを確認した．

4.2 Q&A サイトに対する調査

4.2.1 調査方法

調査対象の Q&A サイトとスレッド件数を次に示す．

- Stack Overflow[*2]：スレッド件数 1,798 件
- Server Fault[*3]：スレッド件数 581 件
- Super User[*4]：スレッド件数 125 件

Stack Overflow は，開発者を対象にしたプログラミングに特化した Q&A サイトである．また，それから派生したシステム管理者のための Q&A サイトが Server Fault であり，いわゆる「パワーユーザ」のための Q&A サイトが Super User である．ミドルウェアの障害については，運用に関する障害が，統合テストやシステムテストなどの開発フェーズでも発生する．このため，管理者やユーザが運用段階で利用する Server Fault や Super User に加え，開発者が主に利用する Stack Overflow も対象に加えている．

これらのサイトの投稿記事は質問と回答がスレッド化されており，各スレッドにはそのスレッドの対象を示すキーワードがタグとして設定されている．まずサーバサイドアプリケーションを構成する LAMP スタックに関連するスレッドを収集するために，“lamp” のキーワードがタグとして設定されているスレッドを，スクリプトプログラムにより収集した．スレッド件数として記載した件数は，収集を実施した 2018 年 4 月 16 日時点においてこの方法により収集したスレッドの総数であり，2008 年 8 月 14 日から 2018 年 4 月 16 日までのスレッドが含まれていた．

収集したスレッドの内容は障害に関するものだけではなく多岐にわたる．単純なものだと，LAMP スタックの意味についてや，インストール方法についてなど，障害に関連しない質問も多く含まれた．運用中の障害に関連する質問としては，次のような質問が含まれていた．

(1) 問題発生時の原因特定方法に対する質問
　問題切り分けのためのログの確認方法や，コマンド出力を利用した原因特定方法に関する議論．
(2) パフォーマンスチューニングに関する質問
　チューニングのための設定ファイルの記述方法に関するベストプラクティス．

そこで収集したスレッドを対象に，目視によりこのような質問が含まれるスレッドを抽出した．その結果，Stack Overflow のスレッド 35 件，Server Fault のスレッド 26 件，Super User のスレッド 3 件を抽出した．

次に，レイヤごとに障害によって表出する障害状況を分類したカテゴリを作成し，収集したスレッドをそのカテゴリに分類した．分類の結果，カテゴリに含まれるスレッドが存在しない場合は，そのレイヤと障害状況をキーワードにして，Q&A サイトから再度検索を実施した．

次節以降では，障害状況を分類するために作成したカテゴリと，カテゴリごとの分類事象を示す．

4.2.2 障害状況分類のためのカテゴリ

運用中の負荷変化により発生する障害は，一般的に次のリソースの負荷増大に起因すると考えることができる [*5]．

(1) メモリ使用量
(2) CPU 利用率
(3) ネットワークコネクション
(4) ストレージアクセス

まず LAMP それぞれのレイヤにおいてこれらの負荷増大がどのような障害状況として表出するかを，4.2.1 節で抽出したスレッドの内容から調査した．その結果，負荷変化により発生するパフォーマンス障害により表出する障害状況は，OS の場合（L レイヤ）の負荷変化とそれ以外の場合（AMP レイヤ）の負荷変化で大きく異なっていた．

OS 上では各リソースに対する負荷の増加が，サーバプロセスの kill（メモリ不足による），応答性能の劣化（CPU 利用率増加による），TCP 接続失敗（ネットワークコネクション増加による）というように，1 対 1 で障害状況として直接的に表出する．一方 OS の上で個別のプロセスとして動作する AMP レイヤには，どのリソースの負荷増加もサーバプロセスに対する応答という間接的な形でしか表出しない．すなわちクライアントから観測される事象としてはサーバに対する実行の失敗（タイムアウト）のみであり，サーバ内の各レイヤで観測される事象としては同時接続数や実行時間制限の超過による接続の切断といった，リソースの負荷に対する間接的な影響による事象であった．

そこで，次のようにカテゴリを作成した．

[*2] https://stackoverflow.com/
[*3] https://serverfault.com/
[*4] https://superuser.com/

[*5] ここに挙げたリソースは，VMware や OpenStack 等の仮想化環境で，負荷軽減のために調節できるリソースを参考にしている．

- L レイヤ（OS）
  - ① メモリ使用量増加による事象
  - ② CPU 利用率増加による事象
  - ③ ネットワークコネクション増加による事象
  - ④ ストレージアクセス増加による事象
- A レイヤ（Web サーバ）
  - ⑤ 同時接続数超過による事象
  - ⑥ 実行時間制限超過による事象
- M レイヤ（データベース）
  - ⑦ 同時接続数超過による事象
  - ⑧ 実行時間制限超過による事象
- P レイヤ（実行エンジン）
  - (-) 同時接続数超過による事象
    ⟶ 同時並列実行数が Web サーバへの同時接続数に依存するため，⑤ と同様の事象とみなした．
  - ⑨ 実行時間制限超過による事象

4.2.3 障害状況の分類の結果

4.2.1 節で抽出したスレッドをカテゴリに分類した．カテゴリに含まれるスレッドが存在しない場合には，カテゴリを表すキーワード（例えば，A レイヤであれば “apache”）と障害状況（例えば，同時接続数超過であれば，“max number of connections”）を指定し，Q&A サイトから再度検索を実施し得られたスレッドを追加した．各カテゴリに対して分類したスレッドを次に示す．

① メモリ使用量増加による事象 (L レイヤ)
　クライアント数増加によりサーバのメモリを使い切ることで，Apache プロセスが kill される事例 [*6] と，MySQL プロセスがシャットダウンされる事例 [*7] ．

② CPU 利用率増加による事象 (L レイヤ)
　クライアント数増加によりプロセス数が増加し，サーバの CPU 利用率が増加するとともに，多数プロセスによりファイル I/O が増加しクラッシュした事例 [*8] ．

③ ネットワークコネクション増加による事象 (L レイヤ)
　TCP 接続数が TCP ソケットで待ち受けられるキューの長さを超過し，セッション確立時に SYN パケットが再送され，応答速度が低下する事例 [*9] ．

④ ストレージアクセス増加による事象 (L レイヤ)
　ディスク領域を使い切って実行が失敗する事例 [*10] と，ファイル I/O 増加によりクラッシュする事例．後者は ② に分類した事例．

⑤ 同時接続数超過による事象 (A レイヤ)
　クライアント数増加により，同時接続数上限の設定値を超過した事例 [*11] ．

⑥ 実行時間制限超過による事象 (A レイヤ)
　サーバからのダウンロード実行時間が，タイムアウトの設定値を超過した事例 [*12]．

⑦ 同時接続数超過による事象 (M レイヤ)
　DB 接続数が設定最大値よりも超過し，DB 実行が失敗する事例 [*13] ．

⑧ 実行時間制限超過による事象 (M レイヤ)
　取得データサイズが大きく，クエリ実行が DB 実行時間制限を超過して DB 実行が失敗する事例 [*14] ．

⑨ 実行時間制限超過による事象 (P レイヤ)
　ストリーミングアプリケーションで対象ファイルが大きく実行時間制限を超過した事例 [*15] と，プログラムロジックの処理が実行時間制限を超過した事例 [*16] ．

4.3 障害状況の再現による確認

各スレッドに記載された障害状況を再現させ，スレッドの記載を参考に実際の障害原因特定手順を確認した．

障害状況の再現は，Intel Xeon プロセッサ E5-2609 v3 (1.90GHz, 6 コア)，メモリ 64GB, ストレージ 1.8TB（SAS HDD，回転数 10krpm，転送速度 12Gb/秒）のサーバ用コンピュータ上に VMware ESXi 6.7 をインストールし，その上に仮想マシン（VM）としてサーバ用 VM とクライアント用 VM を構築した．

各種調査会社・業界団体の市場調査資料 [15][16] によると，国内でも海外でもサーバ仮想化が進み，仮想サーバ上に数台程度の VM を構築し，それらにサーバ環境を構築し一つのシステムとして動作させることが一般化している [*17]．そのような環境でのサーバ間呼び出しを想定し，同一の仮想サーバ内に構築したクライアント用 VM からサーバ用 VM へ大量のリクエストを送信し，障害状況を再現した．

作成した 2 台の VM に 1 コアの CPU と 2GB のメモリと 10 GB のハードディスク領域を割り当て，仮想ネッ

[*6] “VPS Server LAMP - Error establishing database connection”: https://stackoverflow.com/questions/39794456/
[*7] “MySQL server keep on crashing”: https://serverfault.com/questions/517921/
[*8] “High CPU usage resulting in server crash”: https://serverfault.com/questions/418247/ https://serverfault.com/questions/417909/
[*9] “Ubuntu LAMP server busy at peak hours”: https://serverfault.com/questions/726874/
[*10] “phpmyadmin login redirects to same login page”: https://stackoverflow.com/questions/7252456/
[*11] “AH00161: server reached MaxRequestWorkers setting, consider raising the MaxRequestWorkers setting”: https://stackoverflow.com/questions/36924952/
[*12] “504 Gateway Time-out The server didn’t respond in time. How to fix it?”: https://stackoverflow.com/questions/26742580/
[*13] “Sometime display Database connection Error, while loading 100 user in Jmeter. Why?”: https://stackoverflow.com/questions/34531236/
[*14] “Error while sending QUERY packet”: https://stackoverflow.com/questions/30753674/
[*15] “Settings for uploading large files (video/mp4)?”: https://stackoverflow.com/questions/1071415/
[*16] “Fatal error: Maximum execution time of 30 seconds exceeded”: https://stackoverflow.com/questions/5164930/
[*17] 調査資料 [15] では仮想サーバの 4 倍の VM が，[16] では物理サーバの 1.9 倍の OS 稼働台数があることを報告している．

トワーク経由で両方の VM を直接接続した．サーバ上に用意した LAMP それぞれのソフトウェアは次の通りである．

- L : CentOS Linux release 7.5.1804 (64bit)
- A: Apache 2.4.6
- M: mysqld 5.5.56
- P: PHP 7.0.30

そしてサーバ上にアプリケーションとして Wordpress 4.9.6 を配置し，クライアントからサーバの Wordpress コンテンツや静的な HTML コンテンツへの取得を実施した．各障害状況を再現させた手順と結果を次に示す．なお紙面の都合上，障害状況 ① の一部についてのみ，障害時に発生したエラーログの一例を **付録 A.1** に載せる．

① メモリ使用量増加による事象 (L レイヤ)

【再現手順】 DB 呼出しを含む Wordpress コンテンツ取得を複数クライアントから同時実行した．クライアント数を増加させ，サーバプロセス数と，サーバプロセスが利用する総メモリ消費量を増大させた．

【結果】 サーバプロセスが強制終了され，HTTP のサーバ内部エラーが発生した．なお強制終了されるプロセスはタイミングに依存して決定した．

Q&A サイトのスレッドにあるように DB プロセスが強制終了された場合は，付録 A.1 のログが出力された．OS ログに DB サーバプロセス（mysqld）の強制終了が，DB ログに DB サーバプロセスの強制終了によるクラッシュが，Web サーバのエラーログに DB 呼び出しの失敗による PHP プログラム実行のエラー終了が記録された．

別の状況として Web サーバプロセスが強制終了された場合は，OS ログに Web サーバプロセス（httpd）の強制終了が，DB ログに DB サーバプロセスのメモリ不足による再起動が，Web サーバのエラーログに DB 呼び出しの失敗による PHP プログラム実行のエラー終了が記録された．

② CPU 利用率増加による事象 (L レイヤ)

【再現手順】 CPU に負荷をかけた状態で，クライアントからの HTML 取得，PHP 実行，DB 呼び出しを含む Wordpress コンテンツの取得を実施した．なお，CPU 負荷は sar コマンドで確認し，負荷が 0 % の時と，99 % の時で比較した．

【結果】 呼出しにエラーは発生せず，呼び出し時間に有意差は見られなかった．このため，CPU 利用率増加では障害が発生しなかった [*18]．

③ ネットワークコネクション増加による事象 (L レイヤ)

【再現手順】 HTML コンテンツ取得をクライアントから同時実行させた．クライアント数を増加させて，同時接続数を増大させた．

【結果】 セッション確立時に SYN パケットが破棄され再送が行われ，HTML コンテンツの取得時間が増加した（接続数が 1000 の時，通常で 100ms オーダーの取得時間が，最長で 100sec になった）．

④ ストレージアクセス増加による事象 (L レイヤ)

【再現手順】 ディスク領域を使い切った状況として，ファイルシステムのディスク残容量を無くした状態でファイルを作成し書き込みを実施する PHP プログラムを呼び出した．また別の状況として，ファイルシステムの inode を使い切った状態で，同様の PHP プログラムを呼び出した．

【結果】 ディスク残容量が無い状態では，ファイルが作成されるが内容を書き込むことができなかった．ファイルオープンや内容書き込みの命令実行自体は成功する（後者については，0 バイトの書き込みを実施）ため，ログに何も記録されなかった．inode を使い切った状態では，ファイル作成に失敗し Web サーバのエラーログにファイル作成失敗が記録された．

⑤ 同時接続数超過による事象 (A レイヤ)

【再現手順】 Apache の最大同時接続数を設定した状態で，クライアントから HTML コンテンツ取得を同時実行させ，クライアント数を増加させた．

【結果】 呼出しにエラーは発生しないが，通常で 100ms のオーダーの取得時間が最長で 10sec 以上になった．また，Web サーバのエラーログに，最大同時接続数を超過したことが記録された．

⑥ 実行時間制限超過による事象 (A レイヤ)

【再現手順】 実行時間の長い Perl スクリプトを作成し，そのスクリプトをクライアントから呼び出した．

【結果】 HTTP の “504 Gateway Timeout” によるエラーが返却され，タイムアウトしたことが Web サーバのエラーログに記録された．

⑦ 同時接続数超過による事象 (M レイヤ)

【再現手順】 MySQL の最大同時接続数を設定した状態で，DB 呼出しを含む Wordpress コンテンツ取得をクライアントから同時実行させた．クライアント数を増加させて，同時接続数を増大させた．

【結果】 アプリケーションから DB 接続ができず，その結果として実行結果が変わる（WordPress の場合は，エラー画面によって HTTP エラーを返却するが，どう扱うかはアプリケーション実装に依存）．MySQL ではアプリケーションからの接続を拒否する正常な動作であるため，ログに何も記録されなかった．

⑧ 実行時間制限超過による事象 (M レイヤ)

【再現手順】 DB 接続から DB 実行までの処理に，MySQL の DB 実行のタイムアウト設定よりも時間

[*18] 4.2.3 節 ② で挙げたスレッドでは，多数クライアントからのアクセスにより CPU 負荷が上昇し，同時にアクセスされるプログラムのファイル書き出しによりファイルシステムの inode が不足して障害が発生している．この状況については ④ で再現した．

を要する PHP アプリケーションを作成し，クライアントから呼び出した．次に，実行時間の長いクエリを実行する PHP アプリケーションを作成し，クライアントから呼び出した．

【結果】 DB 接続から DB 実行までの処理に時間を要する場合は，アプリケーションからの DB 接続が失われ実行が失敗した．そして，実行失敗が Web サーバのエラーログに記録された（ただし実行失敗をどう扱うかは，アプリケーション実装に依存 *19）．クエリ実行が長い場合は，応答時間が長くなるがアプリケーション実行は成功し，ログに何も記録されなかった．

⑨ 実行時間制限超過による事象 (P レイヤ)

【再現手順】 実行時間の長い PHP プログラムを作成し，クライアントから呼び出した．

【結果】 PHP 実行が途中で終わり HTTP エラーが返却された．Web サーバのエラーログにも PHP の実行エラーが記録された．

5. 考察

5.1 障害発生とログ記録の依存関係

4.3 節によると，LAMP で運用中にパフォーマンス変化に起因しアプリケーション実行に障害が発生する際に，障害とその状況を記録したログとの関係は次のようになった（なお，結果的に障害が発生しない障害状況②は除外した）．

① メモリ使用量増加による事象 (L レイヤ)
A レイヤのプロセスや M レイヤのプロセスに障害を発生させたことが，L レイヤのログに記録される．
後者の場合は，DB 呼び出しを実施するプログラムが実行されれば，A レイヤのログにも記録される．

③ ネットワークコネクション増加による事象 (L レイヤ)
Web サーバへの接続以前の段階で遅延が発生しており，ログへ記録されない．

④ ストレージアクセス増加による事象 (L レイヤ)
ファイル作成失敗が，A レイヤのログに記録される．
ファイル書き込みの失敗は，ログへ記録されない．

⑤ 同時接続数超過による事象 (A レイヤ)
A レイヤのログに記録される．

⑥ 実行時間制限超過による事象 (A レイヤ)
A レイヤのログに記録される．

⑦ 同時接続数超過による事象 (M レイヤ)
ログへ記録されない．

⑧ 実行時間制限超過による事象 (M レイヤ)
DB 接続からクエリ実行の一連の処理が実行時間制限を超過した場合，A レイヤのログに記録される．
DB クエリ実行時間の遅延は，ログへ記録されない．

⑨ 実行時間制限超過による事象 (P レイヤ)
A レイヤのログに記録される．

このように，これらは次の 3 つに分類できる．

- 障害発生レイヤのログに記録される
- 別レイヤのログに記録される
- ログへ記録されない

図 1 に示した階層構造において，Web サーバ以上の階層に配置されるレイヤ（PHP や，その上で動作するフレームワーク）に発生した障害については，Web サーバ（A レイヤ）のログに記録された．また M レイヤは A レイヤとは独立して動作し，M レイヤの設定による制限により実行に障害が発生した場合でも，M レイヤではログには記録されなかった．M レイヤを A レイヤから呼び出した際に M レイヤの呼び出しに失敗した場合は，A レイヤのログに障害が記録された．また OS レイヤ （L レイヤ）のログには，その上の階層である A レイヤ，M レイヤのプロセスの実行に障害が発生した場合のみ記録された．

5.2 ログコリレーションを利用した原因推定

5.1 節で示したログファイルには，障害発生時だけでなく平常時にも運用状況に関する情報が記録される．L レイヤのログファイル（標準で “/var/log/messages”）にはデーモンプロセスの稼働状況が定期的に記録される．A レイヤでは通常はアクセスログとエラーログを分けて運用する．エラーログ（標準で “/var/log/httpd/error_log”）にはアクセスに失敗したログが記録される．公開されている Web サイトであれば，Web サイトへの攻撃を行おうとする攻撃者が定期的にディレクトリリスティング（ディレクトリのインデックスを表示させ，ファイル一覧を取得すること）を実施したり，良くありがちなファイル名に対するファイル取得を試みたりすることが多く，失敗した場合はそのエラーがログとして記録されている．

このようにどちらのログにも，定常状態において定期的にログ記録が蓄積される．障害発生時には，5.1 節で示したログが，それらのログ記録に混入し追加的に注入され記録される．定常状態での定期的なログ記録では同じような内容のログが記録されるが，障害発生時に追加的に記録されるログの内容はそれとは異なる．

アプリストアのレビューよりアプリの不具合の発生を検出する研究 [17] では，Jensen-Shannon ダイバージェンスを利用してレビューのトピックを分類し，定常状態から外れたトピックを検出している．ログ記録に対しても同様の方法によりトピック分類を行うことで，定常状態と外れたトピックが記録されたことを検知できると考える．

5.1 節で示した各障害状況では，障害発生時に**表 1** のように障害に付随する事象を観測できる．

*19 Wordpress の場合は，再接続をするためエラーにはならず，単純に応答時間が長くなる．

表 1 障害発生に対して観測される事象と原因の関係

#	障害原因レイヤ	HTTP Stat. ※	L レイヤログ有無	A レイヤログ有無	推定される障害原因
①	L	500	有	無	A レイヤプロセス障害
		500	有	有	M レイヤプロセス障害
③		-	無	無	ネットワーク接続数増加
④		500	無	有	ファイル作成失敗
		-	無	無	ファイル書き込み失敗
⑤	A	-	無	有	同時接続数超過
⑥		504	無	有	実行時間制限超過
⑦	M	-	無	無	同時接続数超過
⑧		-	無	有	実行時間制限超過
		-	無	無	DB クエリ遅延
⑨	P	500	無	有	実行時間制限超過

※ エラーを示すコードのみを記載．表中の各コードの意味は次の通り．
500: Internal Server Error　504: Gateway Timeout

ここで，障害状況に対して HTTP のエラーコードと各レイヤのログ出力の有無をパターンとして定義し，障害状況の識別に利用することを考えると，表 1 に示すように，障害状況 ① と ⑥ についてはパターンにより原因を一意に推定可能である．一意に推定できない場合でも，次のように追加的な手順の実施により原因を推定可能と考える．

- [HTTP エラー無，L レイヤログ無，A レイヤログ無]
  まず障害状況 ③ については，netstat コマンドで SYN パケットが破棄されたかどうかを確認することで判別できる．そうでない場合，ディスク残容量を確認することで，障害状況 ④ を判別できる．残りのうちの障害状況 ⑦ については，DB クエリに失敗している状態である．対象のアプリケーションの実行結果が DB 取得情報により構成されるならば，正常に動作している状態での実行結果と比較して，応答するメッセージサイズに有意差が生じていることが期待できる．これによって，障害状況 ⑦ を判別でき，どれにも当てはまらない場合が 障害状況 ⑧ である．
- [HTTP500 エラー，L レイヤログ無，A レイヤログ有]
  ディスクの残り inode を確認することで，障害状況 ④ と障害状況 ⑨を判別できる．
- [HTTP エラー無，L レイヤログ無，A レイヤログ有]
  エラーメッセージ発行元が PHP であれば障害状況 ⑧ ，そうでなければ障害状況 ⑤ であると判別できる．

従って，まずログファイル間のログ発生有無の相関（ログコリレーション）とエラーコードに基づき一次的に推定し，分類不可能なものは他の手段によって二次的に推定を行うことで，LAMP スタックに発生した障害に対して原因を推定可能であると考える．原因推定の流れを，**図 2** に示す．まず，運用中にレスポンスタイムに対するベースライン分析や，ステータスコードの変化により障害発生を検出する．そして障害が発生したアプリケーションに対応するエラーログやシステムログを取得したのち，ログ変化量や，トピック分類を実施した結果として得られた定常状態のトピックから外れたログの記録変化量から，表 1 に従い障害原因の一次的な推定を実施する．推定できなかった場合には，応答メッセージサイズの変化とコマンド実行結果を利用して，二次的な原因推定を実施する．これにより，原因がどのレイヤに起因するものかを分類できる．

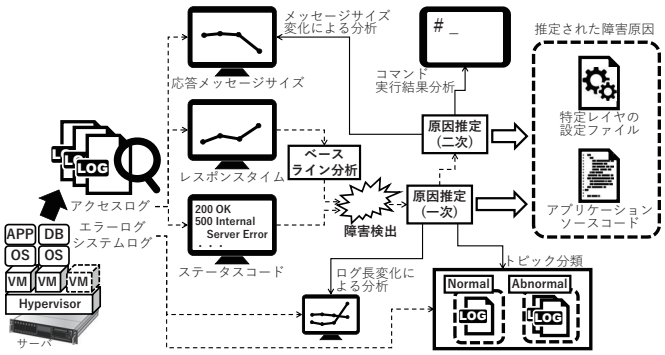

図 2 LAMP スタック障害原因推定の流れ

文献 [13] では，各レイヤのコードに対してコードスライスを実行しスタック間で依存関係のあるオプションを推定し，障害発生時に関連オプションを探すことを支援する方法を提案している．この方法を利用するためには LAMP のミドルウェアのソースコードも含め解析し依存関係を抽出する必要があるが，運用時にミドルウェアのソースコード解析実施を前提とすることは現実的ではない．本研究での調査の結果，ソースコード解析ではなくログ発生関係の分析のみである程度の原因推定が可能であることが分かった．今後は原因を推定する手法を提案する予定である．

5.3 妥当性への脅威

5.3.1 内的妥当性

4.2.1 節で述べたように，Q&A サイトの約 2,500 件のスレッドから運用中の障害に関連するスレッドを目視で抽出した．目視での抽出の過程で，見落としの可能性を否定できない．また 4.2.3 節では，カテゴリに該当するスレッドが無かった場合に，追加でスレッドを探して補完した．しかし，この検索はカテゴリに含まれると考えたキーワードに依存しており，検索結果として得られたスレッドに影響を与えている可能性もある．

5.3.2 外的妥当性

5.2 節で示した障害原因の推定では，障害発生前後でログの増加傾向やトピックが異なることを前提にしている．4.3 節で示したログ発生状況の観察の結果として障害原因の推定が行える可能性を示したが，ログ発生状況は利用するソフトウェア実装に依存する．LAMP スタックに限定すればそれほど大きな差異はないが，ミドルウェアのバージョンが異なると大きな差異が生じる可能性がある．

また L レイヤのログファイルに記録する他のミドルウェア（デーモン）や，A レイヤのログファイルに記録するアプリケーションによっては，本研究で検出対象としたい障害発生時とは異なるタイミングでログを記録し，そのことが障害原因の推定に影響する可能性もある．ただしその場

合でも定常状態で突発的にログ記録が増加することは考えにくいため，このことによる影響は大きくないと考える．

6. 結論

本研究では，Q&A サイトに対する調査と障害状況の再現結果により，LAMP のソフトウェア構成要素に対して障害発生原因とそれにより現れる障害状況がある程度限定されることがわかった．そしてこの関係を利用することで，サーバサイドアプリケーションの障害発生時に，障害原因を特定できる可能性があることがわかった．今後はこの結果を利用した障害原因推定手法を提案する予定である．

謝辞

本研究の成果の一部は，科研費基盤研究 (C)17K00110，2018 年度南山大学パッヘ研究奨励金 I-A-2 の助成による．

参考文献

[1] Seagate Technology LLC: SeaTools, https://www.seagate.com/jp/ja/support/downloads/seatools/ (参照 2018-11-01).
[2] 永井崇之，名倉正剛：迅速な危機回復を目的とする大規模環境向け障害原因解析システム，情報処理学会論文誌，Vol. 54, No. 3, pp. 1109–1119 (2013).
[3] EMC Corporation: Automating Root-Cause Analysis: EMC Ionix Codebook Correlation Technology vs. Rules-based Analysis (White paper), https://www.emc.com/collateral/software/white-papers/h5964-automating-root-cause-analysis-wp.pdf (参照 2018-11-01).
[4] Hitachi Data Systems Corporation: Hitachi Data Systems Enhances IT Operations Analyzer with Expanded Monitoring Capabilities (Press Release), https://www.hitachivantara.com/en-us/news-resources/press-releases/2011/gl111103.html (参照 2018-11-01).
[5] 坂下幸徳，工藤　裕，名倉正剛，草間隆人：大規模クラウドデータセンターの運用管理コストの削減を可能とする IT リソース管理技術，日立評論，Vol. 94, No. 4, pp. 54–57 (2012).
[6] 松田晃一，目黒達生：情報システムの障害状況　2017 年前半データ，SEC journal 第 50 号，Vol. 13, No. 2, pp. 52–58 (2017).
[7] 松田晃一，目黒達生：情報システムの障害状況　2017 年後半データ，SEC journal 第 52 号，Vol. 13, No. 4, pp. 64–71 (2018).
[8] Zaman, S., Adams, B. and Hassan, A. E.: A Qualitative Study on Performance Bugs, *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories*, MSR '12, pp. 199–208 (2012).
[9] Chen, P., Qi, Y., Zheng, P. and Hou, D.: CauseInfer: Automatic and distributed performance diagnosis with hierarchical causality graph in large distributed systems, *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, pp. 1887–1895 (2014).
[10] Anbarasi, M. S. and Vasanthi, B.: Discovery of Web pattern from web logs files using enhanced graph grammar approach, *2017 International Conference on Information Communication and Embedded Systems*, ICICES, pp. 1–6 (2017).
[11] Tong, J., Ying, L., Hongyan, T. and Zhonghai, W.: An Approach to Pinpointing Bug-Induced Failure in Logs of Open Cloud Platforms, *2016 IEEE 9th International Conference on Cloud Computing*, CLOUD, pp. 294–302 (2016).
[12] Jiang, H., Li, X., Yang, Z. and Xuan, J.: What Causes My Test Alarm? Automatic Cause Analysis for Test Alarms in System and Integration Testing, *2017 IEEE/ACM 39th International Conference on Software Engineering*, ICSE '17, pp. 712–723 (2017).
[13] Sayagh, M., Kerzazi, N. and Adams, B.: On Cross-stack Configuration Errors, *2017 IEEE/ACM 39th International Conference on Software Engineering*, ICSE '17, pp. 255–265 (2017).
[14] Wikimedia Foundation, Inc.: LAMP (software bundle), https://en.wikipedia.org/wiki/LAMP_(software_bundle) (参照 2018-11-01).
[15] ITCander, Ltd.: The year to June 2017 $76b server market primed for big growth, http://www.itcandor.com/server-q217/ (参照 2018-11-01).
[16] 一般社団法人電子情報技術産業協会：IT プラットフォーム市場動向及び 2017 年度サーバ出荷実績，https://home.jeita.or.jp/page_file/20180531091235_ZuC7soxIL3.pdf (参照 2018-11-01).
[17] Gao, C., Zeng, J., Lyu, M. and King, I.: Online App Review Analysis for Identifying Emerging Issues, *2018 IEEE/ACM 40th International Conference on Software Engineering*, ICSE '18, pp. 48–58 (2018).

付　　録

A.1 障害状況①でのログ出力例

- OS ログファイルの抜粋

```
Sep 20 14:45:47 localhost kernel: [<ffffffff816ac508>]
     page_fault+0x28/0x30
...(omitted)...
Sep 20 14:45:47 localhost kernel: lowmem_reserve[]: 0 0
     0 0
...(omitted)...
Sep 20 14:45:50 localhost kernel: Out of memory: Kill
     process 29483 (mysqld) score 31 or sacrifice child
Sep 20 14:45:50 localhost kernel: Killed process 29483 (
     mysqld) total-vm:1205976kB, anon-rss:12692kB, file-
     rss:0kB, shmem-rss:0kB
```

- DB ログファイルの抜粋

```
180920 14:46:04 InnoDB: Starting crash recovery from
     checkpoint LSN=3659796
InnoDB: Restoring possible half-written data pages from
     the doublewrite buffer...
```

- Web サーバのエラー用ログファイルの抜粋

```
[Thu Sep 20 14:45:19.211458 2018] [:error] [pid 31662]
     [client 10.40.4.91:39252] PHP Warning: Error while
     sending QUERY packet. PID=31662 in /var/www/html/wp-
     includes/wp-db.php on line 1924
```