小テストの負荷に着目した Moodle 用 Web サーバの性能比較

齊藤 智也 [1)], 王 躍 [1)], 久長 穣 [1)], 西村 世志人 [1)], 末長 宏康 [1)], 金山 知余 [1)],
大平 康旦 [1)], 爲末 隆弘 [1)], 江口 毅 [1)], 今岡 啓治 [1)], 岡田 耕一 [1], 多田村 克己 [1)]

1) 山口大学 情報基盤センター

info-cc@ml.cc.yamaguchi-u.ac.jp

Performance Comparison between Web Servers for Moodle: Focusing on Simultaneous Quiz Load

Tomoya Saito[1)], Yue Wang[1)], Yutaka Hisanaga[1)], Yoshito Nishimura[1)],
Hiromichi Suenaga[1)], Chiyo Kaneyama[1)], Yasuaki Ohira[1)], Takahiro Tamesue[1)],
Tsuyoshi Eguchi[1)], Keiji Imaoka[1)], Koichi Okada[1], Katsumi Tadamura[1)]

1) Center for Information Infrastructure, Yamaguchi Univ.

概要

山口大学情基盤センターでは授業支援システムとして Moodle を運用しているが，以前のシステムではバックエンド側のデータベース・システムの性能不足が顕著であった．特に，100 名以上の学生が一斉に小テストを受験する時にはデータベースの書き込みエラーが発生し，一部の学生の解答が保存されない問題が生じていた．新たなデータベース・システムの導入に伴い，我々は Apache JMeter を用いて多人数が一斉に小テストを受験する動作を模倣し，Moodle システムの応答時間を計測することとした．しかし，Moodle システムの性能は HTTP サービス・ソフトウェアによって異なることが示唆されている．そこで我々は，一斉小テストにおける解答送信時の応答時間に着目し，各種 HTTP サービス・ソフトウェアの性能を比較した．

1 はじめに

山口大学情報基盤センターは授業支援システムとして Moodle [1] を運用している．Moodle は通常の授業のほか，各種の行事・研修会等にも活用されている．

山口大学の Moodle システムは，フロントエンド側の Web システムとバックエンド側のデータベース・システムから構成される．システムを構成するサーバはすべて，大学内のプライベート・クラウド上に構築された仮想サーバである．以前のシステムでは 2017 年頃より，データベースの性能不足が顕著となった．特に，100 名以上の学生が一斉に小テストを受験する時にはデータベースの書き込みエラーが発生し，一部の学生の解答内容が記録されない問題が生じていた．当時のデータベース・システムでは，複数のサーバ間でデータを同期させる仕組みとして，同期型レプリケーションを採用していた．同期型レプリケーションはデータの整合性の維持や耐障害性に優れている反面，データの書き込み性能が低い．Moodle システムの利用者の増加，並びにデータベースへの書き込み処理を伴う利用形態の増加に伴い，本学のサーバ環境では十分な性能を得られなくなった．そこで我々は，2019 年 10 月に，非同期型レプリケーションを採用した新たなデータベース・システムを導入した [2].

国内における新型コロナウィルス感染症の拡大により，2020 年度には Moodle システムの利用科目数，及び同時利用者数が著しく増加した．そこで我々は，2020 年 5 月に，Web サーバの増強を行った．図 1 は，現行の Moodle システムの構成を示している．Web サーバは 3 台から 6 台に増強され，個々の Web サーバの仮想 CPU は 4 コアから 8 コア，仮想メモリは 4GB から 32GB にそれぞれ増強されている．

2019 年 10 月のデータベース・システムの移行に際し，我々は Apache JMeter [4] を用いた負荷テストを行った．多人数が一斉に小テストを受験する動作を模倣するための JMeter テストプランを作成し，Moodle システムの応答時間を計測することとした．しかし，Moodle システムの性能は，Web サーバとして採用する HTTP サービス・ソフトウェアの種類よって異なることが示唆されている [3]．そこで我々は，一斉小テストにおける解答送信時の応答時間に着目し，各種 HTTP サービス・ソフトウェアの性能を比較した．

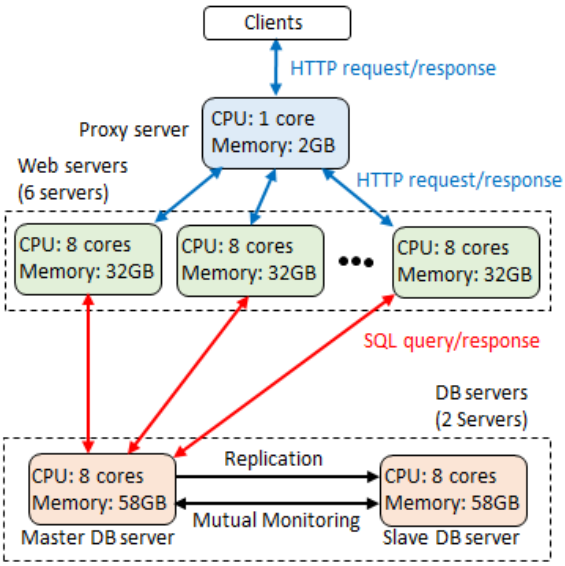

図 1: 現行の Moodle システム

2　JMeter テストプラン

2.1　テストプランの作成

JMeter は，クライアント・サーバシステムの負荷テスト及び性能計測を行うための Java アプリケーションである [4]．JMeter は，予め指定された実施計画（テストプラン）に基づいて各種プロトコルのリクエストを送信し，サーバから返信されるレスポンスを受信する．Moodle 等の Web サイトの負荷テストにおいては HTTP リクエストを送信する．Moodle には「JMeter Test Plan Generator」が付属している [5]．JMeter を活用した Moodle サイトの負荷テスト事例の多くは，このツールによって生成されたテストプランを活用している [6–8]．これらのテストプランでは，JMeter により生成された各々のスレッド（利用者の Web ブラウザに相当）は生成直後から順次，ログイン，コースの表示，ページの表示，フォーラム内の記事の表示，記事への返信の投稿，及びログアウトといった操作を間断なく実行し，処理を終了する．JMeter は予め指定された時間内にすべてのスレッドを生成するが，生成後に実際の処理が開始されるまでの時間には予期せぬばらつきが生じる．また，スレッド数（即ち同時利用者数）の増加に伴い，ログインやコースの表示といった前半の処理における応答時間にばらつきが生じる．応答時間のばらつきの累積により，スレッド間で処理の進捗状況に差異が生じる．コース内に配置された教材へのアクセス開始時刻のばらつきは数十秒を超える場合があり，多数の利用者が特定の教材を一斉に活用する条件下における負荷計測には適していない．特定の教材の活用に関する負荷を計測するため，独自のテストプランを用いて Moodle サイトの負荷を計測した事例も報告されている [10, 10, 11]．しかし，いずれにおいても，各スレッドが生成直後から間断なく処理を進めるテストプランが採用されている．そのため，特に JMeter のスレッド数が多い時には教材のアクセス時刻に大きなばらつきが生じ，その教材の同時利用に関する負荷を計測しているとは言えない．

本稿では，多数の利用者が一斉に小テストを受験する条件下における負荷テストを対象としている．そのため，スレッドの進捗状況を同期させる仕組みを加えた独自のテストプランを作成し，使用することとした．

最初に，山口大学の Moodle サイトにアクセスし，ログイン，共通教育科目のコースへのアクセス，小テストの受験，及びログアウトといった一連の操作を行った．JMeter が内蔵するプロキシを経由して Moodle サイトにアクセスすることにより，一連の操作において生成された HTTP リクエストを記録し，テストプランの原型とした．続いて，テストプランから不要な HTTP リクエストを削除した．例えば，Moodle のキャッシュに配置される JavaScript ファイルや CSS ファイル等の URL は，その時点での一時的なものである．負荷テストにおいてはこれらの URL が存在せず，エラーメッセージが返信される．これらのファイルは Web ページのレイアウトや外観の調節に使用され、各種操作の成否には影響を及ぼさない．このような HTTP リクエストはテストプランから削除した．解答を送信する HTTP リクエストでは，パラメータとして sesskey，attempt，及び qubaid が必要である．これらの値は，ユーザ認証，サイトホーム，小テストの概要といった各 Web ページに含まれている．そのため，Web ページの中からこれらの値を抽出して変数に格納する動作をテストプランに追加した．また，テストプラン中の各 HTTP リクエストの URL 及びパラメータの記述を修正し，前述した各変数の値を使用するようにした．

最後に，実行時パラメータの読み込み，スレッドの作成，外部ファイルからのユーザ情報の読み込み，スレッド間の進捗状況の同期といった各種の動作を追加した．

テストプランを作成後，前述した共通教育科目のコースをテスト環境（3.1 節を参照）の Moodle サイトにコピーし，テスト用コースとした．このコースにテスト用の学生ユーザを登録し，テストプランを実行可能とした．

2.2 テストプランの動作

前節で作成したテストプランの動作を以下に示す．

(1) 指定された数のスレッドが 60 秒間で生成される．
(2) 生成されたスレッドは，Moodle サイトへのログイン，テスト用コースの表示，小テストへのアクセスを行う．
(3) 小テストの概要ページにおいて，各スレッドは他のすべてのスレッドがこの段階に到達するまで待機する．
(4) 各スレッドは 5 秒以内でランダムな時間だけ待機した後，小テストの受験を開始する．
(5) 各スレッドは受験開始から 5 分間待機する．
(6) 各スレッドは小テストの解答内容を保存した後，すべてのスレッドがこの段階に到達するまで待機する．
(7) 各スレッドは 30 秒以内でランダムな時間だけ待機した後，小テストの解答を送信し，レビュー・ページ（採点結果等）を受信する．
(8) 各スレッドは 60〜120 秒の間でランダムな時間だけ待機した後，Moodle からログアウトする．

上記 (3) 及び (6) の処理では同期タイマ（Synchronized Timer）を使用した．同期タイマが設置された箇所では，指定された数のスレッドがその箇所に到着するまで待機し，その数のスレッドが一斉に次の処理を開始する．本研究ではすべてのスレッドの到着を待ち合わせ，一斉に次の処理に進むように設定した．学生が受験開始ボタンをクリックする時刻のばらつきは最大 5 秒と仮定した．学生による解答の送信は試験終了時刻の直前に集中するが，混雑等のトラブルを予想して早めに解答を送信する学生もいると考えられる．そのため，解答送信時刻のばらつきは最大 30 秒と仮定した．負荷テストに使用した小テストは，多肢選択形式が 5 問，穴埋め形式が 8 問の計 13 問から構成されている．穴埋め問題の解答欄は 31 個であるため，小テスト全体では 36 個の解答欄が配置されている．また，この小テストではすべての問題が 1 つの Web ページに一括して表示される．

3 負荷テスト

3.1 テスト環境

図 2 に，性能評価に用いたテスト環境を示す．フロントエンド側の Web システムは，1 台のプロキシ・サーバ及び 3 台の Web サーバから構成されている．クライアント PC では，JMeter がテストプランに従って HTTP リクエストを送信し，レスポンス（Web ページやエラーメッセージ等）を受信する．プロキシ・サーバは，Web サーバへの HTTP リクエストの振り分け，並びにレスポンスの転送を行う．加えて，HTTPS 通信の暗号化／復号化も担当する．なお，一斉小テストにおける HTTP サービス・ソフトウェアの性能に焦点を当てるため，キャッシュ・サーバ等は実装していない．

プロキシ・サーバ及び Web サーバは，大学内のブレード・サーバ上に仮想サーバとして構築されている．これらの仮想サーバの諸元は次の通りである．

- CPU: 4 コア（合計 8.0 GHz 相当）
- メモリ: 4 GB
- ディスク: 100 GB
- ネットワーク・インターフェース: 10 Gbps
- OS: CentOS 7.7 1908

バックエンド側のデータベース・システムは，現行の Moodle システムのものをそのまま利用している．システムは 2 台のデータベース・サーバから構成され，マスタとスレーブが明確に区別されている．SQL クエリはマスタ DB サーバにおいて処理される．また，データベース・システムでは，相互監視デーモン（keepalived）を用いたフェイルオーバーの仕組みを導入している．

データベース・サーバの諸元を以下に示す．

- CPU: 8 コア（合計 16.0 GHz 相当）
- メモリ: 58 GB
- ディスク: 250 GB
- ネットワーク・インターフェース: 10 Gbps
- OS: CentOS 7.7 1908
- データベース管理システム: MariaDB 10.3.13
- 相互監視デーモン: keepalived 1.3.5

また，仮想サーバの基盤となるブレード・サーバの諸元は次の通りである．

- CPU: Intel Xeon E4-24700 2.3 GHz（8 コア）×2
- メモリ: 128 GB
- ディスク・インターフェース: FC-SAN 8 Gbps
- ネットワーク・インターフェース: 10 Gbps
- 仮想化ハイパーバイザ: VMWare ESXi 5.5.0

プロキシ・サーバ及び Web サーバは，2 台のブレード・サーバに分散配置されている．各々のブレード・サーバでは他のシステムも動作しているため，それら

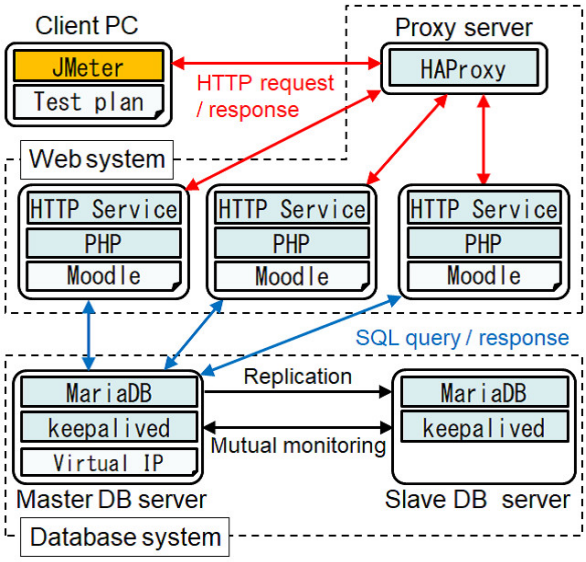

図 2: テスト環境

の仮想サーバとの間で相互排他的にブレード・サーバのリソースを利用する．データベース・サーバは 1 台のブレード・サーバに配置され，この 2 台の仮想サーバのみでブレード・サーバを占有している．

テスト環境で使用した各種ソフトウェアのバージョン等は，JMeter 5.0.2，HAProxy 2.0.6，PHP 7.4.3，及び Moodle 3.8.2 である。Web サーバ 1 台あたりの最大接続数は 500，プロキシ・サーバの最大接続数は 2000 とした．

HTTP サービス・ソフトウェアとして Apache 2.4.6 [12], Lighttpd 1.4.54 [13], 及び nginx 1.16.1 [14] を使用し，これら 3 種類の HTTP サービス・ソフトウェアの性能を比較した．また，PHP の実行形態として PHP FPM（FastCGI Process Manager）を採用し，Web サーバごとに起動可能な PHP の最大プロセス数は 300 とした．

Apache MPM（Multi Processing Module）にはいくつかの種類が存在するが，今回は event を採用した．また，Apache の設定における主要なパラメータ値を以下に示す．

```
TimeOut                 60
KeepAlive               On
MaxKeepAliveRequests    1000
KeepAliveTimeout        5
StartServers            10
MinSpareThreads         50
MaxSpareThreads         50
ThreadLimit             64
ThreadsPerChild         25
MaxRequestWorkers       150
MaxConnectionsPerChild  0
```

続いて，Lighttpd における主要なパラメータ値を以下に示す．

```
server.max-keep-alive-idle      3
server.max-keep-alive-requests  16
server.max-read-idle            60
server.max-write-idle           360
```

最後に，nginx における主要なパラメータ値を以下に示す．

```
keepalive_timeout           5
client_header_timeout       60
client_body_timeout         60
reset_timedout_connection   on
send_timeout                60
fastcgi_connect_timeout     10
fastcgi_read_timeout        60
fastcgi_send_timeout        360
```

これらのパラメータ値は，負荷テストを通じて，応答時間が小さくなるように試行錯誤的に調節した結果として得られたものである．

3.2 テスト結果

小テストの同時受験者数の影響について把握するため，JMeter のスレッド数を変動させながら負荷テストを実施した。プロキシ・サーバが負荷分散の対象とするサーバ数を変更することにより，サーバ数の影響についても計測した．Web サーバごとの最大接続数を 500 としたため，サーバが 1 台のときは，スレッド数が 500 以下の範囲のみ計測した．また，今回の負荷テストでは 10 回の負荷テストにおける計測結果の平均値をグラフに示している．

図 3 及び 4 はそれぞれ，スレッド数の変動に対する平均応答時間を示している．ここで応答時間とは，スレッドが解答を送信してからレビュー・ページ（採点結果等）を受信するまでに経過した時刻である．多人数が一斉に小テストを受験する状況においては一般に，受験する学生ユーザへの応答時間を一定の範囲内に収めることが求められる．JMeter は受信したコンテンツの解釈や画面への表示は行わないが，実際の Web ブラウザでは，コンテンツの解釈と表示に要する処理時間が応答時間として加算される．本稿では，Web ブラウザの処理を含む応答時間の上限を 10 秒，Web ブ

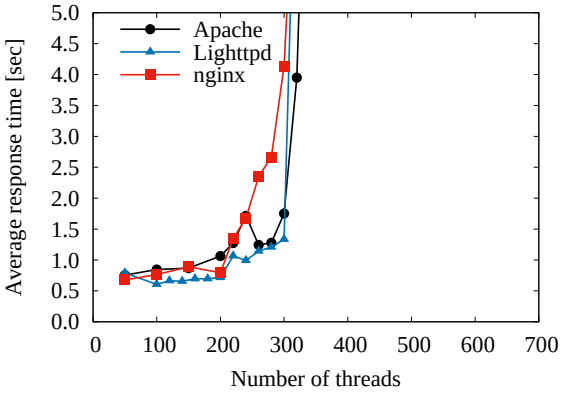

図 3: スレッド数に対する平均応答時間（1 サーバ）

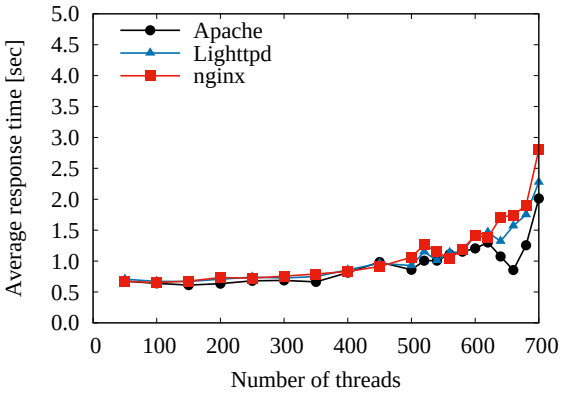

図 4: スレッド数に対する平均応答時間（3 サーバ）

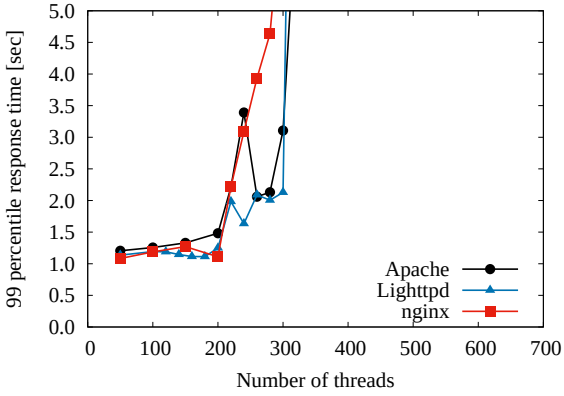

図 5: スレッド数に対する 99% 応答時間（1 サーバ）

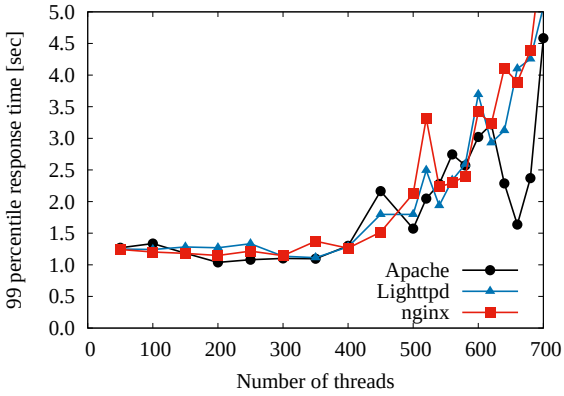

図 6: スレッド数に対する 99% 応答時間（3 サーバ）

ラウザにおける処理時間を 5 秒と仮定した．従って，JMeter における応答時間の上限は 5 秒とした．

図 3 及び 4 より，すべての HTTP サービス・ソフトウェアに共通した傾向として，スレッド数の増加に伴い，平均応答時間は増加する．

図 3 より，Web サーバが 1 台の環境下においては，スレッド数が 300 を超えると平均応答時間は急激に増加する．これは，今回のテスト環境では PHP FPM の最大プロセス数を 300 に設定したためである．例えば，400 個のスレッドから一斉に HTTP リクエストが送信された場合，300 個のリクエストについては即座に処理が開始されるが，残りの 100 個のプロセスについては処理を待機する状態となる．そのため，スレッド数が 300 を超えると応答時間は著しく増加する．

PHP FPM の最大プロセス数を大きな値に設定することにより，スレッド数の増加に伴う応答時間の増加を抑制することが出来る．ただし，同時に起動可能なプロセス数の増加は，Web サーバのメモリの不足を招く恐れがある．今回のテスト環境では，Web サーバの仮想メモリは 4GB としている．接続スレッド数が 300 を超えると Web サーバの仮想メモリが不足し，スワップ領域（仮想ディスクの一部）が使用される．スワップ領域のアクセス速度はメモリに比べて極めて低速であるため，スワップ領域を使用しているプロセスが増加すると，Web サーバの応答時間は著しく増加する．今回の負荷テストでは，小テストの問題文及び選択肢はテキスト情報のみで構成されている．図表などの画像情報を含む小テストを使用する環境においては，さらに記憶容量の大きなメモリが必要になる．

図 3 及び 4 より，3 種類のサービス・ソフトウェア間では平均応答時間に大きな差異は見られないことが示された．ただし，Lighttpd は Apache 及び nginx に比べてグラフのゆらぎが小さい．これは，Lighttpd は他の 2 つのサービス・ソフトウェアに比べて負荷テストの各回における応答時間のばらつきが小さいためである．

図 5 及び 6 はそれぞれ，スレッド数の変動に対する 99 パーセント応答時間を示している．ここで 99 パーセント応答時間とは，各スレッドにおける応答時間をサンプル値とした場合の 99 パーセント値である．理論的な最大応答時間は，最小応答時間 （システムのサービス時間に相当）と学生数（負荷テストにおいて

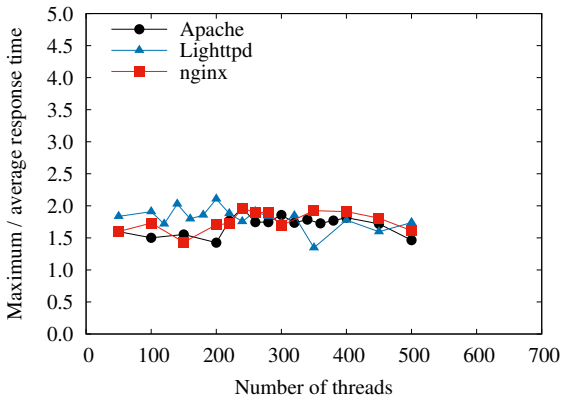

図 7: スレッド数に対する最大-平均比率（1 サーバ）

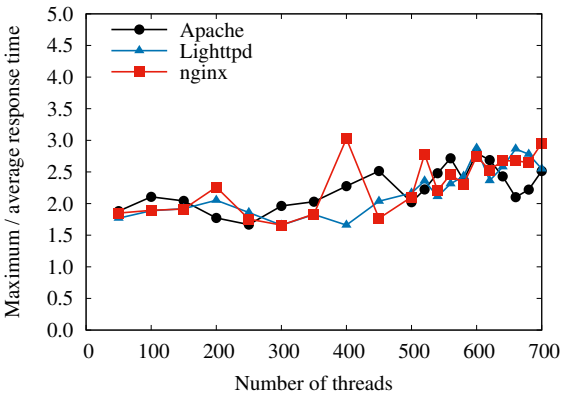

図 8: スレッド数に対する最大-平均比率（3 サーバ）

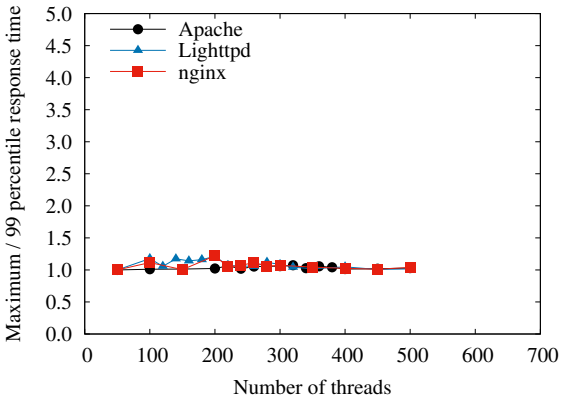

図 9: スレッド数に対する最大-99% 比率（1 サーバ）

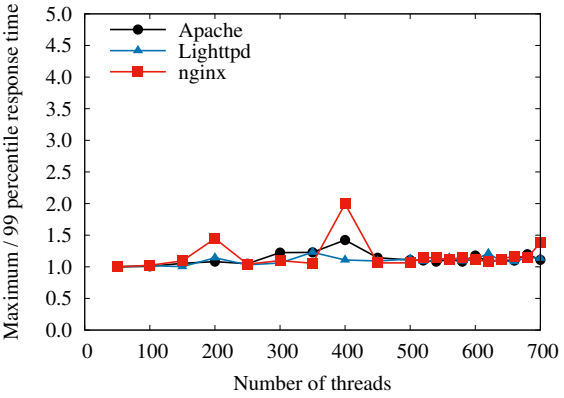

図 10: スレッド数に対する最大-99% 比率（3 サーバ）

はスレッド数）を乗算した値である．テスト環境における最小応答時間は 0.5 秒程度であったため，例えば，スレッド数が 500 のときの最大応答時間は 250 秒程度となる．実際のシステムにおける最大応答時間は，まず，学生間での解答送信時刻の重複度合いにより，最小応答時間から理論的な最大応答時間の間のさまざまな値をとり得る．この値に，その時々におけるネットワーク転送の遅延時間，及び他の仮想サーバの影響によるリソース割り当ての待ち時間などが加算されるため，正確な計測・予測は困難である．そのため，本稿では 99 パーセント遅延時間を性能の指標とした．

図 5 及び 6 より，すべてのサービス・ソフトウェアに共通した傾向として，スレッド数の増加に伴って 99 パーセント応答時間は増加する．99 パーセント応答時間が 5 秒未満の範囲で最も大きなスレッド数を最大収容人数と仮定すると，いずれのサービス・ソフトウェアにおいても，サーバ数の増加に伴って最大収容人数は増加する．

図 5 より，平均応答時間に関する計測結果と同様に，Web サーバが 1 台の環境下においては，スレッド数が 300 を超えると応答時間が著しく増加する．これは，PHP の最大プロセス数を 300 に設定したためである．

図 3~6 より，サービス・ソフトウェア間では応答時間に大きな差異は見られないことが示された．ただし，平均応答時間に関する計測結果と同様に，Litghttpd は他のソフトウェアに比べて各回の負荷テストにおける応答時間のばらつきが小さいことが分かった．

続いて，応答時間のばらつきについて考察する．図 7 及び 8 は，スレッド数の変動に対する最大-平均比率の特性を示している．ここで最大-平均比率とは，最大応答時間を平均応答時間で除した値である．

図 7 及び 8 より，すべてのサービス・ソフトウェアに共通の傾向として，最大-平均比率はスレッド数の増減の影響を受けにくいことが分かる．ただし，Web サーバが 3 台の環境下では，1 台の時に比べてテスト毎の計測値のばらつきが大きく，グラフのゆらぎが大きい．各々の Web サーバにおける HTTP リクエストの処理時間には，その時々の負荷状況や他の仮想サーバとのリソースの競合などにより，ばらつきが生じる．処理時間のばらつきは，サーバ数の増加と共に顕著となる．特にスレッド数（同時利用者数）が多い場合にはサーバ間で処理の進行状況の差が大きくなるため，

毎回の負荷テスト毎の応答時間のばらつきが大きくなる．ただし，サービス・ソフトウェア間の比較においては，Lighttpd は最もグラフのゆらぎが小さく，毎回の小テストにおいて安定した性能を提供していることが分かる．

図 9 及び 10 はそれぞれ，スレッド数に対する最大-99% 比率を示している．ここで最大-99% 比率とは，最大応答時間を 99 パーセント応答時間で除した値である．

図 9 及び 10 より，共通の傾向として，スレッド数の増加に伴い，最大-99% 比率はスレッド数の増減の影響を受けにくいことが分かる．また，最大-99% 比率は最大-平均比率に比べて小さい値となるものの，Web サーバの台数の増加に伴い，毎回の計測値のばらつきが大きくなる．

図 7~10 より，サービス・ソフトウェア間では応答時間のばらつきに大きな差異は見られないことが示された．また，Web サーバ数の増加に伴い，スレッド間，すなわち学生間での応答時間のばらつきが大きくなることが示された．試験等において受験者全体により公平なサービスを提供するためには，Web サーバの台数を増やすより，Web サーバあたりのスペックを向上させるほうが望ましい．ただし，少数の Web サーバにリソースを集中させることは耐障害性の低下につながるため，システム管理者はサーバごとのスペックとサーバ数のトレードオフに留意しなければならない．

本節で得られたテスト結果より，今回のテスト環境においては，小テストの回答を送信する際の応答時間は HTTP サービス・ソフトウェアの種類には左右されないことが示された．Web サーバにおいて小テストの解答の受信，正誤の判定，受験記録のデータベースへの書き込み，及びレビュー・ページの生成といった一連の処理を行う場合，PHP プロセスにおける処理時間が応答時間の大半を占める．そのため，小テストの応答時間に最も大きく影響するパラメータは，PHP FPM の最大プロセス数である．Web サーバにおいて CPU のコア数を増強し，PHP FPM の最大プロセス数を大きく設定することによって応答時間の軽減を図ることが出来る．ただし，PHP の同時稼働プロセス数の増加はメモリの不足の原因となるため，同時利用者数が多い環境においてはメモリの増強も必要である．

HTTP サービス・ソフトウェアのパラメータの中では，KeepAlive のタイムアウト時間が Web サーバの性能に最も大きな影響を与える．タイムアウト時間を小さく設定すれば無駄なネットワーク・コネクションの維持に伴う CPU 負荷の増加やメモリの消費を抑制することができ，Web サーバの性能を改善することができる．今回の負荷テストにおいても，タイムアウト時間が大きすぎる場合には応答時間の増加，及び収容可能人数の減少が生じた．ただし，KeepAlive のタイムアウト時間を小さく設定した場合，低速あるいは不安定なネットワーク環境からシステムに接続しているクライアントにおいては頻繁にコネクションの切断と再接続が行われる．そのため，一部の学生においては応答時間が増加する恐れがある．

今回の負荷テストに用いたサービス・ソフトウェアの中では，Lighttpd を採用した場合に，計測結果のばらつきが最も小さくなることが示された．そのため，Lighttpd は他のソフトウェアに比べ，性能の予測やパラメータの調節が容易である．

また，サービス・ソフトウェアのパラメータ値の調節においては，Lighttpd は他のソフトウェアに比べてデフォルト値からの設定変更，及び設定項目の追加が少なかった．今回の負荷テストにおいては，「server.max-keep-alive-idle」の値を変更したのみで，他のソフトウェアと同等の性能を示した．システム管理者が Lighttpd を用いて Moodle 用 Web サーバを構築する場合，この値に加え，サーバのスペックに応じて最大接続数を調節すれば良い．

4　まとめ

本稿では，多人数が一斉に小テストを受験する際の負荷に着目し，各種 HTTP サービス・ソフトウェアの性能を比較した．HTTP サービス・ソフトウェアの性能を評価するためのテスト環境を構築し，JMeter を用いた負荷テストを実施した．負荷テストに際し，一斉小テストの動作をシミュレートするためのテストプランを作成した．HTTP サービス・ソフトウェアとして，Apache，Lighttpd，及び nginx を使用した．3 種類のソフトウェアを Moodle の Web サーバとして採用して負荷テストを行い，それらの性能について比較した．

負荷テストの結果より，各々の HTTP サービス・ソフトウェアのパラメータが適切に設定されている場合には，ソフトウェアごとの性能の差はわずかであることが示された．本稿ではいずれのサービス・ソフトウェアにおいても，PHP FPM を採用した．PHP FPM では，PHP の処理を担当するサービスが OS 上に常駐しており，HTTP サービスと分離されている．多人数による一斉小テストのように PHP における処

理時間が長い用途においては，PHP プロセスの処理時間が応答時間の大半を占める．そのため，特に PHP FPM を採用している環境下においては，HTTP サービス・ソフトウェアの違いによる性能の差が小さくなる．

それぞれのサービス・ソフトウェアにおいて応答時間が小さくなるようにパラメータ値の調節を行った場合，Lighttpd が最もパラメータの変更箇所が少なかった．また，負荷テストにおいては 10 回のテストの平均値を性能として示したが，Lighttpd は各回のテストにおける応答時間のばらつきが小さく，最もグラフのゆらぎが小さかった．そのため，Lighttpd は他のサービス・ソフトウェアに比べ，性能の予測やパラメータの調節が容易であることが分かった．

前述したように，HTTP サービス・ソフトウェアの性能が応答時間に与える影響は小さい．ただし，PHP FPM における最大プロセス数と共に，サービス・ソフトウェアにおける KeepAlive のタイムアウトの値は応答時間の性能にある程度の影響を及ぼす．本稿の執筆段階では，これらのパラメータが Web サーバの性能に及ぼす影響についての詳細な調査・考察は出来ていない．今後の課題として，それぞれの HTTP サービス・ソフトウェアにおけるパラメータ値の影響についての詳細な調査が挙げられる．

参考文献

[1] Moodle: Open-source learning platform, https://moodle.org/ （2020 年 9 月 10 日確認）.

[2] 齊藤智也 他，「小テストの負荷に着目した Moodle 用データベース・システムの構築および性能評価」，日本ムードル協会全国大会（2020）発表論文集，pp.15–21 （2020）.

[3] Performance recommendations, https://docs.moodle.org/39/en/Performance_recommendations （2020 年 9 月 30 日確認）.

[4] Apache JMeter, https://jmeter.apache.org/ （2020 年 9 月 10 日確認）.

[5] JMeter test plan generator, https://docs.moodle.org/37/en/JMeter_test_plan_generator （2020 年 9 月 30 日確認）.

[6] M. Manzo,「Design and Performance Evaluation of a Virtualized Moodle-based E-Learning Environment」, Journal of E-Learning and Knowledge Society，vol.12，no.3，pp.19–30 （2016）.

[7] 桑田喜隆 他，「パブリッククラウドを使った Moodle の構築および運用評価.」，日本ムードル協会全国大会（2018）発表論文集，pp.41–47 （2018）.

[8] 浜元信州 他，「クラウドを利用したログ解析環境の Moodle への適用」，日本ムードル協会全国大会（2019）発表論文集，pp.24–31 （2019）.

[9] 王躍 他，「Moodle 小テスト時の負荷シミュレーションテスト」，情報処理学会研究報告，vol.2010-IOT-10，no.11，pp.1–5 （2010）.

[10] 王躍 他，「OSS に基づいた Moodle サイトのスケーラビリティに関する報告」，情報処理学会研究報告，vol.2011-IOT-14，no.2，pp.1–5 （2011）.

[11] S. Chauhan et al.,「Exploiting Moodle Performance under Various Configuration」, Innovative Systems Design and Engineering，vol.7，pp.1–6 （2016）.

[12] The Apache HTTP Server Project，https://httpd.apache.org/ （2020 年 9 月 30 日確認）.

[13] Lighttpd – fly-light，https://www.lighttpd.net/ （2020 年 9 月 30 日確認）.

[14] nginx，https://nginx.org/ （2020 年 9 月 30 日確認）.