YAMAGUCHI UNIVERSITY

MoodleMoot 2019
2019年2月27日～3月1日

Moodle用データベースのシステム構成に関する検討

山口大学 大学情報機構 メディア基盤センター

齊藤 智也，王 躍，久長 穣，多田村 克己

背景と目的

- 山口大学のMoodleシステムでは，100名を超える受講生が一斉に小テストを実施すると不具合が発生。
  - 過負荷によりページの表示／切り替え速度が大幅に低下。
  - データベースの書き込みエラーが発生して回答が記録されない。
- データベースが最も大きなボトルネックとなっているため，データベースの性能改善が急務。
  - 使用可能なサーバ資源は，余っているブレード・サーバ1台のみ。
  - 障害発生時のフェイルオーバーのため，2台以上のサーバで構成。
- 現状のサーバ資源で構成可能なデータベース・システムについて検討。
  - システムごとに3種類のベンチマークを実施。

使用可能なハードウェア

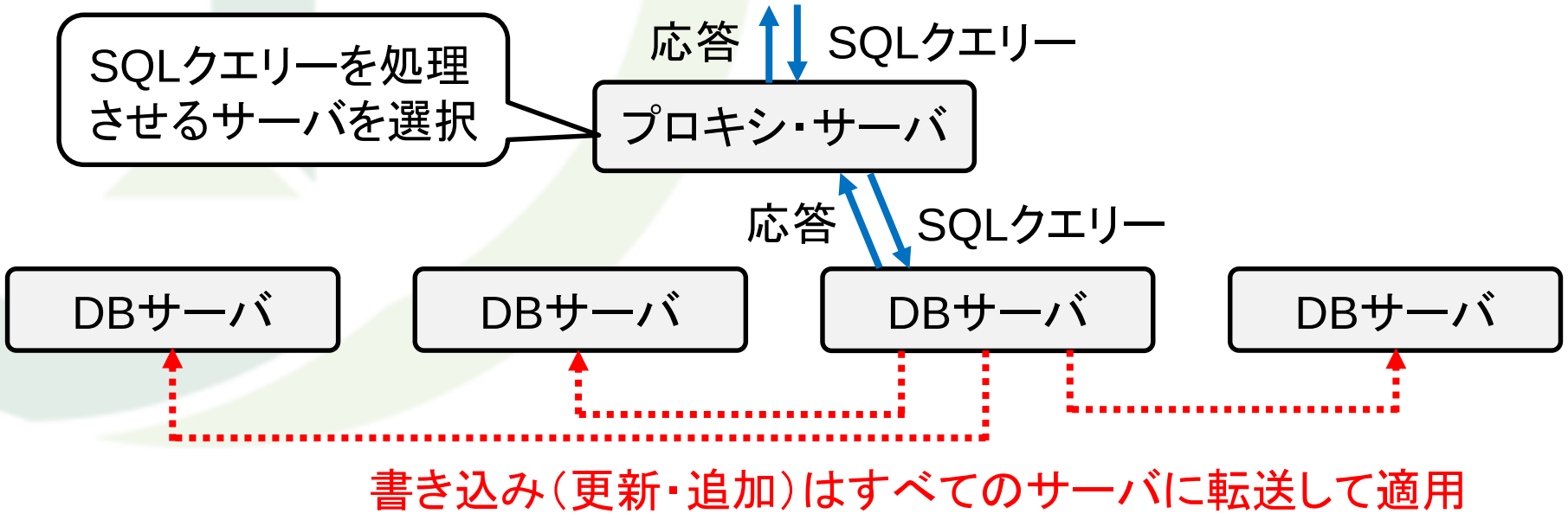
■NEC製のブレード・サーバ

- VMware ESXiを使用して仮想化。
- CPUは5年ほど前のIntel Xeon E5シリーズ。
  - 物理CPUの周波数は3.2GHzで仮想CPUの周波数は2.0GHz
  - 物理コア数は16コア，スレッド数は32。
- メインメモリは128GB(仮想サーバでは118GBまで使用可能)。
- 8Gbpsのファイバチャネルを介してNASと接続。
  - 仮想サーバに使用するディスク(仮想ディスク)はNASから確保。
  - NASに接続されているディスクはハードディスク(10000RPM, SAS)で，RAID 6による障害対策を採用。
- NASの費用も考慮すると，ブレード・サーバ1台あたりの費用は1000万円程度。

現在のシステム構成

■4台のサーバによる仮想同期型の相互レプリケーション。

- いずれのサーバに対しても書き込み（更新・追加）が可能。
- データの更新はすべてのサーバに転送され，適用される。
  - ➢ すべてのサーバで書き込みが完了するまで新たな処理は待機。
  - ➢ 新たな処理は割り振り先のサーバにおいて処理の実行待ちになる。



ベンチマーク
（1）mysqlslapを使用したベンチマーク

■MySQLやMaria DBに付属しているコマンド・ベースのベンチマーク・ツール。

- クライアント数（スレッド数）やクエリーの種類（データの参照，追加，更新，混合），クエリーの発行回数を指定して実行。

■今回の検証では以下の条件で実施。

- クライアントごとのクエリー発行回数は6000回。
- クライアントあたりの平均処理時間（6000回のクエリーの処理が終了するまでの平均時間）を計測。
- 計測の開始前に10万件のデータを追加。
- ベンチマークを5回繰り返し，平均値を計測結果として採用。

■計測値が小さいほど性能が良い。

ベンチマーク
(2)JOINの処理に対するベンチマーク

■Moodleデータベースのログの中から処理時間が長く，比較的頻繁に使用されているSQLクエリーについて調査した結果，JOIN(テーブルの結合)を多く使用する2つのクエリーを発見。

■この2つのクエリーを連続して実行する際の処理時間を計測。

- 100回の繰り返しの後，1回あたりの平均処理時間を結果として採用。
- 計測値が小さいほど性能が良い。

ベンチマーク
(3)TPC-Cに基づいたベンチマーク

■TPC-Cは大規模なオンライン・トランザクション・システム(例えば大規模な電子商取引サイト)の性能を検証するためのベンチマークの規格。

- 個々のトランザクションの処理内容は比較的単純だが，多数のクライアントから休みなくトランザクションが発生し続ける。
- 1分あたりに処理可能なトランザクション数が性能の指標。計測値が大きいほど性能が良い。
- 検証では「tpcc-mysql」(MySQLを対象としたオープンソースのプログラムで，TPC-Cの処理内容をシミュレート)を使用。

■今回の検証では以下の条件を採用。

- 在庫を管理する倉庫数は50(8GB程度のデータ量)。
- 初期データを登録した後に5分間の処理を行い，そこから2時間のベンチマークを実施。

システム構成の検証
(1)MySQLクラスタ

■複数のサーバ(データノードと呼ぶ)にデータを一部分ずつ分散保有させて負荷分散を図る仕組み。

■今回は以下のようなシステムを構成。

- プロキシ・サーバ(仮想CPU×2, メモリは2GB)は2台。1台は予備。
  - SQLクエリーを処理させるサーバ(SQLノード)を選択。
- SQLノード(仮想CPU×3, メモリは4GB)は3台。
  - データノード上のデータを使用してSQLクエリーを処理。
- データノード(仮想CPU×4, メモリは25GB)は4台。
  - 実際のデータを保有。
- MySQL Cluster 7.6を使用。

システム構成の検証
(2)Maria DB Galera Cluster

■現行のMoodleシステムでも採用している仮想同期型レプリケーション。

■今回は以下のようなシステムを構成。

- プロキシ・サーバ(仮想CPU×2, メモリは2GB)は2台。1台は予備。
- データベース・サーバ(仮想CPU×4, メモリは38GB)は3台。
- MariaDB 10.3に付属のGalera Clusterを使用。

システム構成の検証
(3)非同期型レプリケーション

- 1台のサーバがマスターとしてSQLクエリーを処理。
- マスターはスレーブが更新処理を正常に受信して適用したかは確認しない。
  - 更新処理が頻繁に発生した場合にはマスターとスレーブの間で保有するデータの内容にズレが生じる。
  - 同期型／準同期型レプリケーションに比べて書き込み性能が高い。というよりは単一のサーバに比べて書き込み性能が低下しにくい。
- 今回は以下のようなシステムを構成。
  - データベース・サーバ(仮想CPU×9, メモリは8GB)は2台。
  - プロキシ・サーバ(仮想CPU×2, メモリは4GB)は2台。1台は予備。
- Maria DB 10.3を使用。

ベンチマークの結果（クライアント数は100）

■単一のデータベース・サーバ(Maria DB)の結果も掲載。

- 仮想CPUは4コアもしくは9コア、メモリ8GB。Maria DB 10.3を使用。
- Galera ClusterやMySQL Clusterのシステム構成のまま1台だけ使用。

■Galera Clusterでは書き込み速度が極めて低速だったため，TPC-Cベンチマークは実施しなかった。

■MySQL ClusterやGalera Clusterを導入するにはサーバが非力。

- すべてのサーバを別々のブレード上に配置するくらいのリソースが必要。

システム ＼ ベンチマーク	mysqlslap [秒]				JOIN [秒]	TPC-C [tpm]
	参照	追加	更新	混合		
単一サーバ(4コア)	11.375	41.878	41.624	26.111	0.90	2951
単一サーバ(9コア)	9.437	45.524	46.721	26.436	0.30	6135
MySQLクラスタ 7.6	10.997	101.715	16.821	54.987	68.23	15286
Maria DB Galera Cluster	19.872	697.591	753.820	386.587	0.30	
非同期レプリケーション	8.652	42.218	41.174	26.913	0.31	9671

システムの検証
(4)物理サーバ(PC)

■10万円程度の自作PCを使用。
- CPUは第6世代 Core i5で，メモリは8GB。
- SATA-3(6GBps)のSDDを採用。
- Maria DB 10.3を使用。

■1000万円もする仮想サーバ環境よりはるかに高速であるが，クライアント数が増加すると動作が不安定になり，接続エラーが増加。
- 実際の運用に際しては，電源，マザーボード，ネットワーク･インターフェースが強化されているサーバ専用機が必要。

物理サーバと仮想サーバの比較

- 単一サーバと非同期レプリケーション(2台)の性能を計測。
- TPC-Cの結果が大きく異なるのは，仮想サーバではHDDを使用し，物理サーバではSSDを使用しているため。
  - HDDのIOPS(1秒あたりに実行可能な入出力命令の数)は数百程度であるが，SSDのIOPSは数万から数十万程度。
- 物理サーバでは非同期レプリケーションを採用した場合に書き込みの性能が大きく低下。
  - PC向けの4コアのCPUではレプリケーションの処理速度が不足？

システム＼ベンチマーク		mysqlslap [秒]				JOIN [秒]	TPC-C [tpm]
		参照	追加	更新	混合		
物理サーバ (4コア)	単一	6.893	47.385	48.142	32.304	0.30	26236
	非同期	6.900	122.294	123.053	76.691	0.30	24445
仮想サーバ (9コア)	単一	9.437	45.524	46.721	26.436	0.30	6135
	非同期	8.652	42.218	41.174	26.913	0.31	9671

クライアント数の影響については省略

■クライアント数を100から1000まで変動させながら，それぞれのシステムに対してベンチマークを実施したが，時間の都合により今回は省略。

- 最初は40分の発表を予定していたものの，査読者から20分の発表への変更を勧められたので。。。

ベンチマークを実施した結果

■仮想環境による非同期レプリケーションを採用。

■性能面ではSSDを採用した物理サーバを購入できれば良いが，新たな物理サーバを購入する予算がない。

- Moodleシステム自体の予算はゼロのため，大学のプライベート･クラウドの中で余っているサーバしか使用できない。

■メンテナンス面では仮想サーバのほうが有利。

- Moodleの管理者がいずれもサーバ室と離れたキャンパスに勤務しているため，ハードウェアの障害に迅速に対応できない。

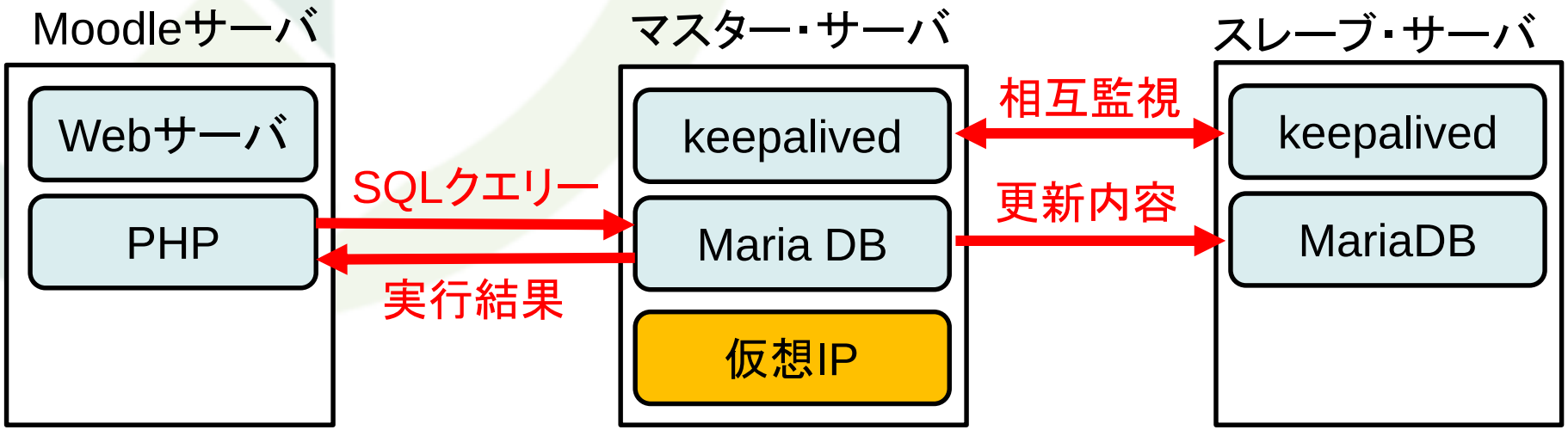
他の仮想サーバの影響

- 同じブレード・サーバ上に他のサーバが存在しているときと，存在しないときで性能に違があるかどうかを検証。
- ベンチマーク結果の中で，「他のサーバあり」の環境では，起動しているだけで特に仕事をしていないサーバが3台存在。
- ベンチマーク結果より，他のサーバが稼働しているだけでデータベース・サーバの性能が低下することが分かった。
  - 仮想サーバが稼働しているだけでCPUリソースを喰っている。

システム ＼ ベンチマーク		mysqlslap [秒]				JOIN [秒]	TPC-C [tpm]
		参照	追加	更新	混合		
他のサーバあり（仮想CPU×9）	単一	9.437	45.524	46.721	26.436	0.30	6135
	非同期	8.652	42.218	41.174	26.913	0.31	9671
他のサーバ無し（仮想CPU×8）	単一	10.141	40.168	38.280	27.145	0.53	18653
	非同期	10.376	70.293	70.189	40.479	0.53	10756

新システムの構成

- 2台の仮想サーバによる非同期レプリケーション。
  - サーバごとの仮想CPUは8コア，メモリは58GB。
- VRRPを使用する相互監視デーモン(keepalived)を使用して，自動的に一方がマスター，もう一方がスレーブとなる。
- マスターが仮想IPを保有し，Moodleサーバは仮想IPを持つデータベースサーバ(マスター)に接続。
- マスターに障害が発生するとスレーブが自動的にマスターに昇格。



Moodleの小テストを用いた動作検証

■JMeter(HTTPリクエストを記録・再生するソフトウェア)を使用して複数の利用者が38問の小テストを一斉に受験する動作を再現し，Moodleシステムの負荷を計測。

- 同時利用者数が300名のときのデータベース・サーバの負荷
  - サーバのロード・アベレージは2.5程度。
  - メモリ使用量はアイドル状態の時からほぼ上昇しなかった。
  - ネットワーク上のデータ転送量は100Mbps程度。ブレード・サーバのNICの転送速度は10Gbpsであるため，1パーセント程度の負荷に相当。
  - 処理時間が5秒を超えるクエリーは発生しなかった。
  - 1秒あたりのクエリーの発行数は2200件程度で，9割が参照処理であった。この条件では，新システムは1秒あたり22000件まで処理可能であるため，10パーセント程度の負荷に相当。

■既存のシステムに比べて性能の改善が期待できる。

まとめ

- 1台のブレード・サーバ上で構成可能なMoodle用のデータベース・システムについて検討。
  - MySQLクラスタ, Maria DB Galera Cluster, 非同期レプリケーション, 物理サーバについてベンチマークを実施。
- ベンチマークの結果により, 2台の仮想サーバによる非同期レプリケーションを採用。
  - 同じブレード・サーバ上に同居する仮想サーバが何も仕事をしていなくても, 稼働しているだけでリソースの競合につながるため, 他の仮想サーバを削除して2台だけで運用。
- JMeterを用いて多数の学生が同時に小テストを受験する際の負荷について計測。
  - 新システムの採用により性能の改善を期待できることが分かった。
  - Moodle 3.0以降では小テストの動作の仕組みがMoodle 2.9までと異なるため, JMeterのシナリオの作成に従来とは違った工夫が必要。

今後の課題

■小テストやコンテンツの閲覧に際する負荷について，より詳細な検証を実施。

- 今回のJMeterを用いた計測では，同時利用者数が150名を超えるとMoodleサーバにボトルネックが生じていたため，複数のMoodleサーバを使用した検証が必要。

■既存のデータベース･システムから新たなデータベース･システムへの移行を進める。

- 2019年10月に移行予定。

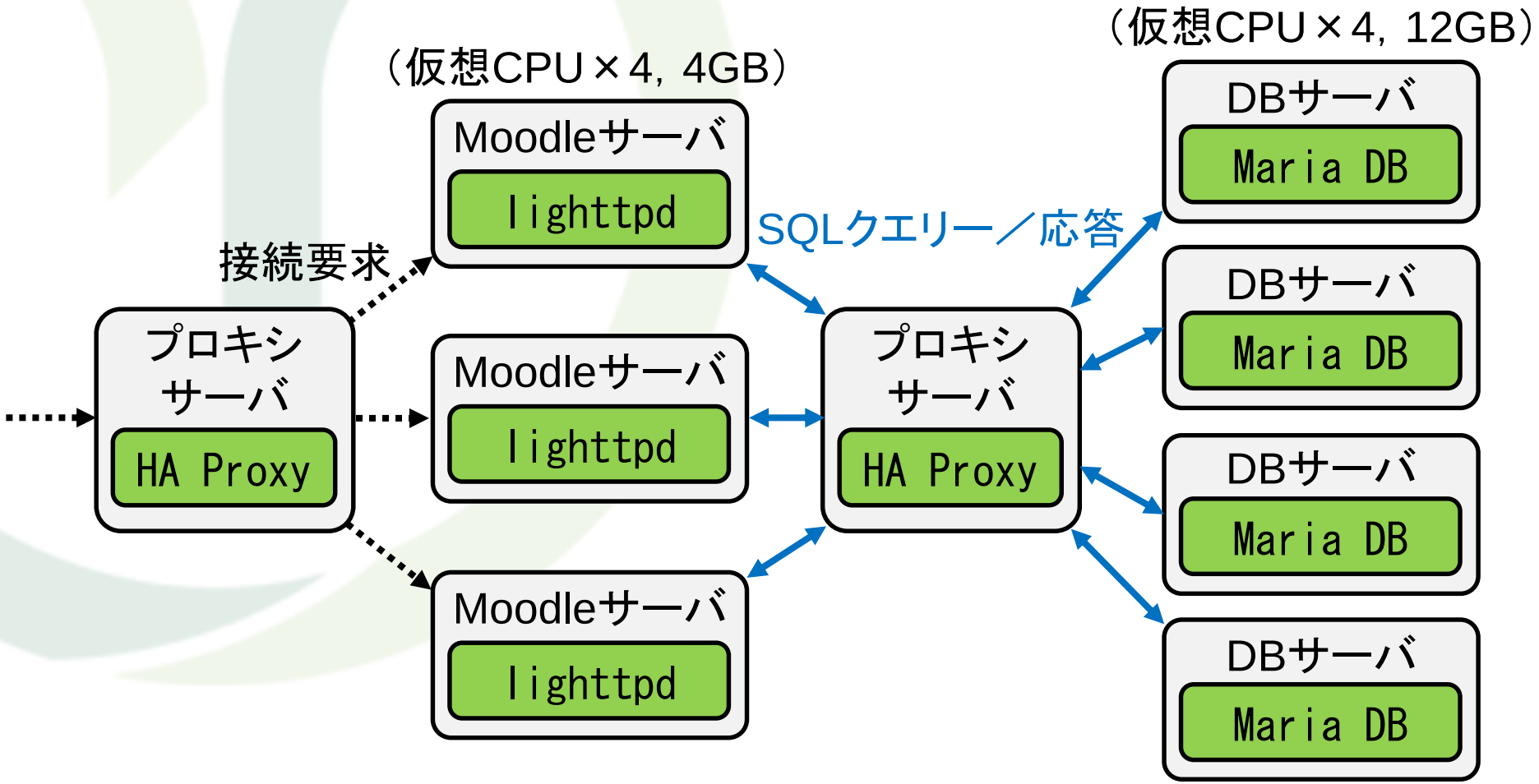
■より一般的で，ベンチマークを実施しやすいJMeterのシナリオとMoodleコースを作成して公開。

- MoodleMoot 2020で発表（？）。

現在のMoodleシステムの構成

■すべてのサーバは仮想サーバ。

■2台のブレード・サーバ上に構成。



クエリーの例（1）

■JOINを複数回使用するもの。

```
SELECT c.id AS course, u.id AS userid, crc.id AS completionid,
ue.timestart AS timeenrolled, ue.timecreated
FROM mdl_user u
INNER JOIN mdl_user_enrolments ue ON ue.userid = u.id
INNER JOIN mdl_enrol e ON e.id = ue.enrolid
INNER JOIN mdl_course c ON c.id = e.courseid
INNER JOIN mdl_role_assignments ra ON ra.userid = u.id
LEFT JOIN mdl_course_completions crc ON crc.course = c.id
AND crc.userid = u.id
WHERE c.enablecompletion = 1 AND crc.timeenrolled IS NULL
AND ue.status = 0 AND e.status = 0 AND u.deleted = 0
AND ue.timestart < '1537445702'
AND (ue.timeend > '1537445702' OR ue.timeend = 0) AND ra.roleid IN (14)
ORDER BY course, userid;
```

クエリーの例(2)

■JOINの他にサブクエリーを使用するもの。

```
SELECT qas.id, qa.questionusageid, qa.questionid, qa.variant,
qa.slot, qa.maxmark, qas.fraction * qa.maxmark as mark
FROM mdl_quiz_attempts quiza
JOIN mdl_question_attempts qa ON qa.questionusageid = quiza.uniqueid
JOIN mdl_question_attempt_steps qas ON qas.questionattemptid = qa.id
       AND qas.sequencenumber = (
            SELECT MAX(sequencenumber) FROM mdl_question_attempt_steps
            WHERE questionattemptid = qa.id
          )
WHERE
quiza.quiz = '15490' AND quiza.preview = 0
AND quiza.state = 'finished'
AND (quiza.state = 'finished'
    AND NOT EXISTS (SELECT 1 FROM mdl_quiz_attempts qa2
                WHERE qa2.quiz = quiza.quiz AND
                qa2.userid = quiza.userid AND qa2.state = 'finished'
                AND qa2.attempt < quiza.attempt))
AND quiza.sumgrades IS NOT NULL
AND qa.slot IN ('1','2','3','4','5','6','7','8','9','10','11','12','13');
```

MySQLクラスタの導入に関する問題点

- 複数のテーブルを結合する処理や，テーブル内の多数のレコードの参照を必要とする処理が低速。
  - 複数のデータノードに分散しているデータを収集してから処理を進めなければならない。
- JOINの問題への対策
  - データを分散していないスレーブ・サーバを準備し，JOINを含む処理はスレーブ・サーバに割り振る。
  - SQLノードやデータノードのスペックの向上。
  - Moodleが発行するSQLクエリーの改善。
- Moodleの独自な改変はバージョン・アップ等の長期的な運用で問題が生じ得る。
- サーバ資源に余裕が無いので，サーバのスペックの向上やスレーブ・サーバの準備は困難。

Galera Clusterの導入に関する問題点

■単一サーバの場合に比べ，参照の処理時間は同程度であるが，追加・更新の処理時間は7倍にもなる。

■検証環境では，ディスクへの書き込み待ちによる負荷の増大が顕著だった。

- すべての仮想サーバで単一のRAIDシステムを使用しているため，ディスクへのアクセスがボトルネックになる。
  - 単一のディスクを複数のサーバで共有している状態に近い。
- HDDを用いてRAIDを構成しているため，IOPS(単位時間あたりに処理可能な入出力命令の個数)が小さい。
  - ディスク1台あたりのIOPSは，SSDが数万から数十万であるのに対し，HDDでは数百程度。

検証を通じて得られた教訓

■データベースでは複数のサーバが同時に高負荷となる。

■処理を抱えた仮想CPUのコア数が物理CPUのコア数を超えると，CPUの割り当て待ち状態が長くなる。

- 仮想CPUが待ち状態にある時間が長くなり，仮想サーバの負荷は小さいように見えるが，処理時間は著しく増大する。

■特に1台の物理サーバ上でデータベース・システムを構成する場合，仮想CPUのコア数が物理CPUのコア数を超えない範囲で仮想サーバを作成しなければならない。

非同期レプリケーションの注意点

■マスターがスレーブへの更新内容の反映を確認しないため，スレーブからの応答を待たなくてすむ反面，新規データの追加が大量に連続すると，スレーブでのデータ更新に遅れが生じる。

■スレーブはマスターの停止を検知した後，未適用の更新データをすべて適用した後にマスターに昇格しなければならない。

- 本システムではkeepalivedを使用しているため，遅れ状態のチェックや更新の適用を待つスクリプトを自前で作成しなければならない。