使用Java平台针对数据库桥接层进行Spring框架可靠性调查

Article *in* Procedia Computer Science · December 2019

DOI: 10.1016/j.procs.2019.11.214

CITATIONS
2

READS
391

2 authors:

Arief Ginanjar
Langlangbuana University
13 PUBLICATIONS **14** CITATIONS

SEE PROFILE

Mokhamad Hendayun
Langlangbuana University
15 PUBLICATIONS **25** CITATIONS

SEE PROFILE

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 161 (2019) 1036–1045

Procedia Computer Science

www.elsevier.com/locate/procedia

第五届信息系统国际会议2019

使用Java平台针对数据库桥接层进行Spring框架可靠性调查

Arief Ginanjar*, Mokhamad Hendayun

University of Langlangbuana, Jl. Karapitan No. 116, Bandung, 40261, Indonesia

摘要

无论是在Web应用程序中还是在桌面应用程序中，都有几种框架可用于简化Java编程环境中的创建应用程序。如果我们更多地关注Java Web框架，那么从2004年开始流行的是Spring框架，尤其是Spring框架的功能，它可以与其他各种框架（例如Hibernate框架，Ibatis或今天的MyBatis框架）结合使用，以及其他一些框架。进行这项研究的目的是比较使用Java编程语言和Spring Framework构建的Web服务应用程序中加载数据的能力，特别是如果与诸如Java数据库连接（JDBC），Hibernate Framework，MyBatis Framework等数据库桥接层结合的使用，以及其他功能。结合Hibernate和MyBatis包含的框架功能具有缓存数据层。性能测试方案是在Spring Framework中创建一个Web服务，然后通过使用第三方代码构建的自定义测试脚本进行访问，并在特定时间段内重复调用它。

2019年第五届信息系统国际会议科学委员会负责的同行评审。

关键词: Spring Framework; Hibernate; Ibatis; MyBatis; JDBC; Cache Engine; Web Service. JSON.

* Corresponding author. Tel.: +62-82-121-633-200.
E-mail address: arief.ginanjar@unla.ac.id

2019年第五届信息系统国际会议科学委员会负责的同行评审。
10.1016/j.procs.2019.11.214

1. 介绍

1.1. 背景

Spring框架是最早在2000年代初期开发并开始用于使用Java平台制作Web应用程序的框架之一。 随着时间的推移，出现了具有各种优势的其他框架，但是Spring框架的流行仍然持续存在，并且一直使用到2018年。这可以从以下信息图形中看出，这些图形说明了Java程序员对某些最受欢迎的框架的兴趣程度 在2004年至2018年之间的世界范围内。

从图1的信息图中，有一些框架在程序员的工作中很普遍地被实现，包括Spring框架，Hibernate，JSF，Struts和MyBatis，这五个框架是所使用的框架中最受欢迎的框架。 由Java程序员编写。

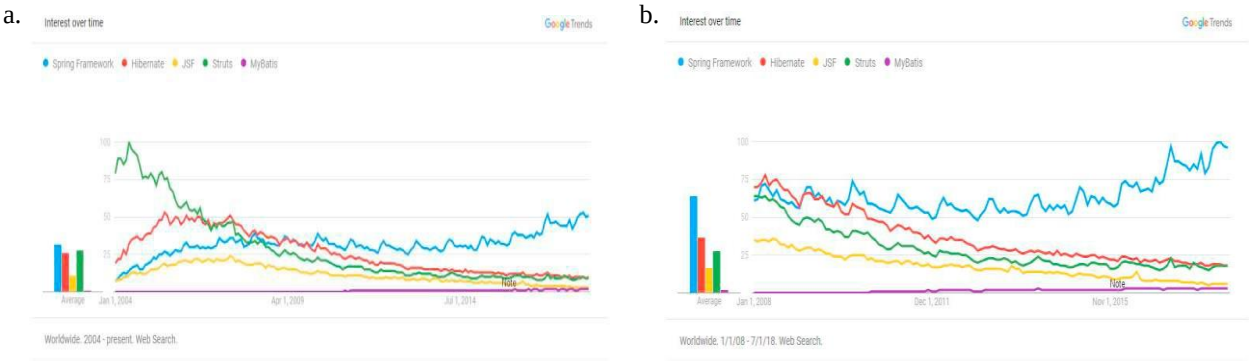

Fig. 1. 2004年至2018年（a）和2008年至2018年世界上几个流行框架喜好程度；（b） (Source: Google Trend).

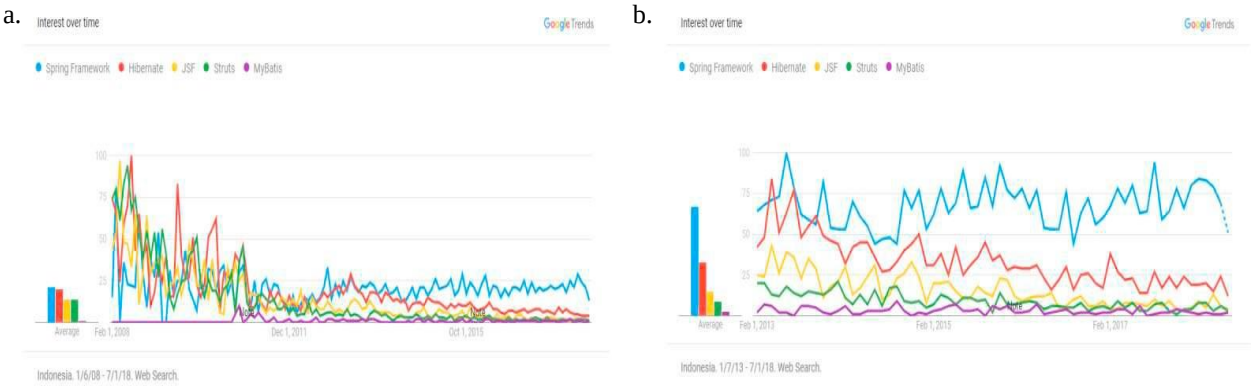

Fig. 2. 从2008年至2018年（a）和2013年至2018年在印度尼西亚几个流行框架的喜好程度；（b） (Source: Google Trend).

在图2-a中，印度尼西亚的Java程序员框架在2008年至2010年之间存在波动，这表明在应用程序工作中使用的框架仍处于试用阶段，但在2011年之后，它开始显示出重要的意义。 程序员倾向于从每个流行的框架中受益并从中受益的趋势。 然后在图2-b中可以看到印度尼西亚的Java程序员在2013年至2018年期间倾向于使用五个框架的趋势。

在基于Java的应用程序开发中实现Spring框架的使用时，通常将其与其他框架结合使用，包括： Java数据库连接（JDBC），带有Hibernate商标的对象关系映射，带有Ibatis或MyBatis商标的Java Query Expose Connection，以及缓存的使用。

该引擎除了存储数据库中存储的数据外，还用于存储临时数据，目的是提高应用程序数据访问速度。

1.2. 研究目标

本研究要达到的研究目标是了解表1中列出的每个标本的可靠性值，并将使用印度尼西亚全国各省，市，地区，街道和乡村的数据名称进行一系列测试，对这些标本进行测试。 将通过以一定强度在json数据URL上加载循环来测试98,457行数据和数据加载性能

Table 1. 将要测试的框架标本

Specimen	Main Framework	Bridging Framework	Cache Framework
1	Spring MVC	JDBC	-
2	Spring MVC	Hibernate	-
3	Spring MVC	Hibernate	L2 Ehcache
4	Spring MVC	MyBatis	-
5	Spring MVC	MyBatis	LRU MyBatis

通过一系列测试，可以期望当每个框架都必须使用负载循环来进行性能测试时，它能够知道每个框架的可靠性。 因此，在选择基于Java的应用程序开发中使用的Spring Framework协作时，它有望成为决策的参考源。

1.3. 研究范围

这项研究仅专注于数据加载网址，并未按照以下技术规范测试插入操作，更新和删除json网址：

- 使用在Microsoft Windows 7操作系统中单独放置的MariaDB 10.1.19版数据库系统，Oracle VM Virtual Box 5.2.6版中的正版64位。
- 在本地主机之间使用网络连接-guest虚拟机。
- 将Microsoft Windows 7正版64位操作系统用于主机环境。
- 使用Netbean 8.2作为集成开发编辑器。
- 使用 Spring Framework version 4.0.1 作为 MVC Framework.
- 使用Spring-json taglib作为json输出的库。
- 使用应用程序容器Apache Tomcat 9.0.12版作为应用程序部署
- 使用java虚拟机1.8.0.162 64 bit 并使用默认设置，不进行任何别的设置，以此作为平台环境的调整

2. 文献综述

2.1. 研究方法

编写本研究报告所使用的方法是使用定量研究方法和原型，以及使用重复过程和系统方法来测试框架性能的过程，其中方法强调要素和组成部分[1 ,2，3]，而 处理顺序如下：

- 文献研究，研究可用作参考的图书馆资源。 图书馆资源可以是书籍，论文，或者讨论Spring框架，Hibernate框架，Ibatis框架，和缓存引擎的网页。

- 分析，描述如何对每个框架的体系结构和编程技术进行分析，以及该技术如何将这些框架中的每一个进行组合。
- 软件设计，将在框架之间进行组合的软件设计，这些框架将基于从分析中获得的结果进行构建。 设计必须符合将要执行的测试方案。
- 软件实现，将基于从设计中获得的结果来开发的实现软件。 此实现将产生包含以下框架组合的软件产品：在要测试的样品中指定。
- 测试和评估，对已构建的软件产品进行测试，然后对所执行的每个测试方案进行性能评估。
- I最终的软件设计和实施阶段以及测试和评估，对软件样本进行测试，当业务流程未满足研究人员的意愿时将进行检查，然后进行标本软件的设计和重新实现。
- 应用程序性能测试，对应用程序通过url循环加载施加压力的能力进行测试过程，该过程针对使用url循环脚本方法和研究人员构建的Java编程脚本对内置到应用程序中的标本框架重复进行。

2.2. 进化原型模型

在研究实施部分中描述的过程阶段涉及原型模型的演化阶段的情况下，该过程可以如图3所示。原型的演化过程包括四个主要过程，即输入，原型过程， 和输出，但在此过程中还必须受到条件的限制； 所构建的每个功能必须满足系统要求中指定的适当要求，并且参与原型方法论过程的人员的能力必须满足最低系统要求。 [4].

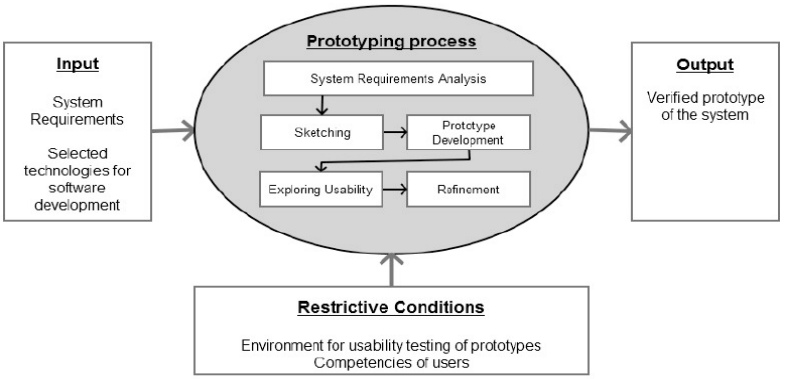

Fig. 3. Evolutionary Prototyping Model [4].

2.3. 标本结构

通过使用面向对象编程原理以及模型，视图和控制器MVC方法的方法，抽象使程序员无需复杂的组件即可开发复杂的思想，而封装使我们无需思考即可专注于软件功能 详细介绍过程的复杂性. [5, 6].

通过将Spring框架，Hibernate框架和MyBatis框架架构组合成一个系统，该系统重叠了每个框架的缺点，因此可以预期在组合这些框架时可以进行并可以优化的测试. [7, 8, 9].

a. b.

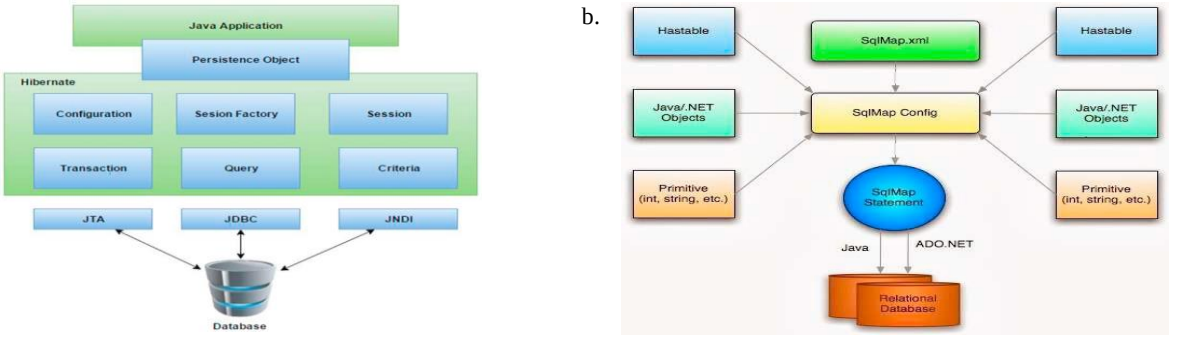

Fig 4. Hibernate Framework Architecture (a) [9]; and MyBatis Framework Architecture (b) [8].

3. 研究方法

3.1. 研究背景

当使用进化原型开发软件时，所执行的步骤从分析，设计和实现，测试和评估开始，并在限制性条件下反复进行，以产生所需的输出。 目的是找到框架的最佳组合，以在java平台中生成Web服务生成器。

3.2. 标本配置

在已经执行的测试样本中，已经应用了技术方面，即： 进行的测试涉及多个系统层，即数据库层，应用程序容器层，逻辑编程层和测试脚本层。 然后，在每个层中都使用了几种配置，以便该层中的每个系统都可以与其他层进行交互。

Table 2. Configuration Implementation at Each *Layer*.

No	Layer	Technology	Configuration
1	Database	MySQL	Default
2	Application Container	Apache Tomcat	Default
3	Logic Programming	Spring Framework Combination	Specifically each specimen
4	Script Pengujian	Java Programming	Specifically each specimen

然后，将每个样本的测试流程存储在来宾OS上安装的虚拟服务器上，然后将Apache Tomcat安装在主机OS上，然后使用NetBean IDE 8.2版运行测试脚本，其插图如图2所示。

表1中显示的每个框架标本还需要其他几个库来支持研究标本中的主要框架，表3中列出了库的详细信息。

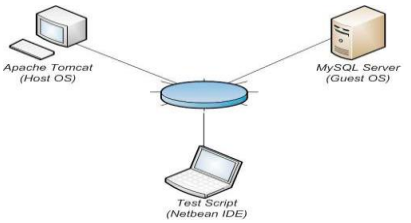

Fig 5. Spring框架数据流体系结构测试了几个数据库桥接层的性能。

从样本列1（该样本是Spring Framework 4.0.1版与Java数据库连接（JDBC）结合使用的测试样本）中可以看出，它需要另一个库，即JSP标准标记库（JSTL），该库具有转换数据的功能 从Java类环境到Java网络环境，然后MySQL-JDBC是专门用于具有MySQL商标的数据库服务器的Java数据库连接（JDBC），然后还辅以JSON Taglib（一种转换工具，用于从JSTL环境转换数据） Javascript对象表示法.

Table 3. Additional Library for Each Specimen.

Library	Specimen				
	1	2	3	4	5
JSTL	√	√	√	√	√
MySQL JDBC	√	√	√	√	√
JSON Taglib	√	√	√	√	√
Persistence JPA 2.1	×	√	√	×	×
AOP Alliance	×	√	√	×	×
Ehcache	×	×	√	×	√

鉴于Persistence JPA 2.1是用作Spring框架和Hibernate框架之间的桥梁的主要支持工具，因此这两个框架之间的数据交换是有效的，而当有一些功能无法实现时，AOP Alliance成为Persistence JPA 2.1的补充功能。 在Persistence JPA 2.1的帮助下，它将由AOP联盟协助满足互操作性需求。 Ehcache是一种基于硬件或软件的技术，该技术通常用作临时数据存储，用于帮助提高应用程序或系统的性能。

缓存技术的应用已被应用程序开发人员广泛使用，以提高访问应用程序数据视图的速度，每个框架在对所管理的数据实施缓存时都有其自己的体系结构，成为下一个研究材料非常有趣 。

3.3. 资料收集

研究人员进行的数据收集是以运行应用程序样本的形式运行的，该应用程序样本由表1所示的框架组成，并以产品名称Apache Tomcat版本9.0.12安装在应用程序容器环境中，然后在测试脚本中使用 如图6所示的Netbean 8.2环境.

```
public static void main(String[] args) throws IOException, JSONException {
    // TODO code application logic here
    long secondStart = System.currentTimeMillis();
    JSONArray jsonArray = readJsonArrayFromUrl("http://localhost:8080/Research-Spring-Jdbc/json.htm");
    for (int i = 0; i < jsonArray.length(); i++) {
        //get the JSON Object
        JSONObject obj = jsonArray.getJSONObject(i);
        Integer id = obj.getInt("id");
        String nama = obj.getString("nama");
        Integer anak = obj.getInt("anak");
        System.out.println("id = "+id+", nama = "+nama+", anak = "+anak);
    }
    long secondFinish = System.currentTimeMillis();
    long elapseTime = secondFinish - secondStart;
    System.out.println(" second "+elapseTime);
}
```

Fig 6. 命令行使用Java编程测试了几个数据库桥接层的spring框架性能。

4. 结果

4.1. 研究结果

在对Spring框架标本和数据库桥接框架相结合的性能进行测试时，发现MyBatis框架与L2 Cache的组合之间出现了积极的趋势，而Hibernate Framework与L2 Cache的组合却出现了消极的趋势。 L2缓存。

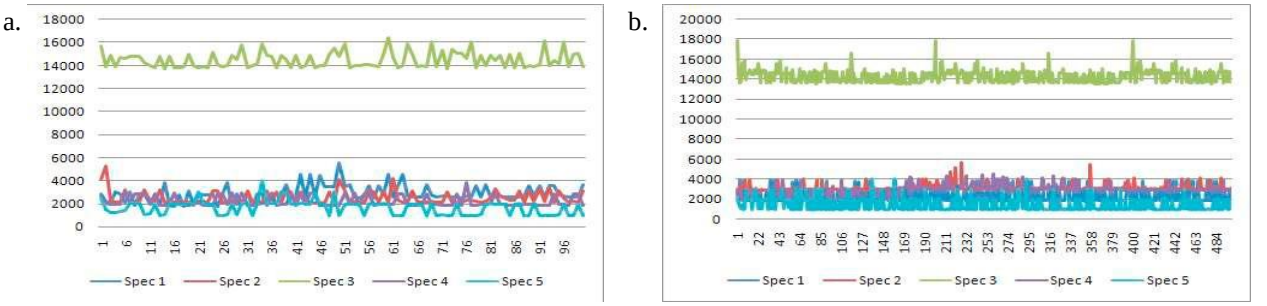

Fig 7. Scenario one test results for 100 hits (a); and 500 hits (b).

从基于预定测试场景进行的测试中，发现当场景1和场景2的命中率均高于1000时，无法执行包含Spring Framework，Hibernate Framework和2nd Ehcache的三个样本 在第三个，第四个，第五个和第六个场景中，使用相同的样本不能跑到500以上。

在上述条件下，作者足以集中于命中100个数据和命中500个数据，目的是使所有标本和场景的观测数据结果保持平衡。 在接下来的讨论中，已显示数据的表示形式已转换为如图7至图12所示的线形图。

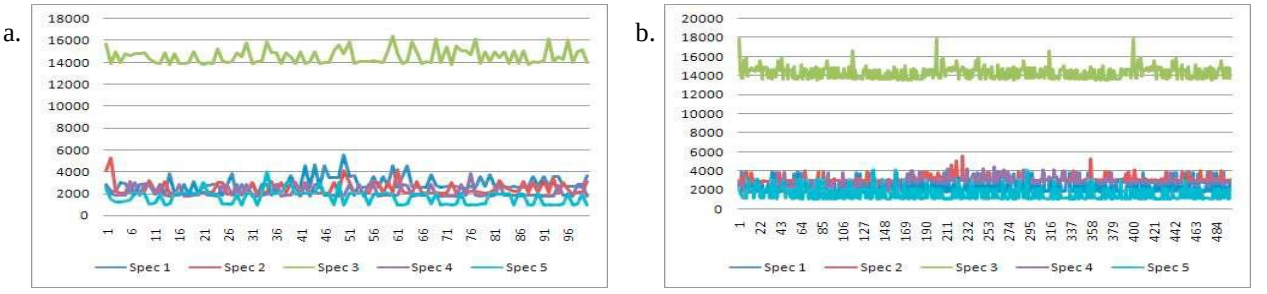

Fig 8. Scenario two test results for 100 hits (a); and 500 hits (b).

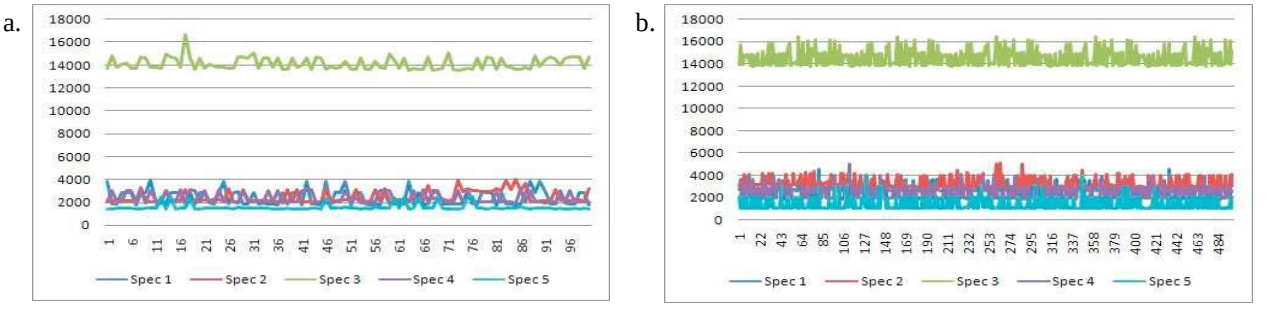

Fig 9. Scenario three test results for 100 hits (a); and 500 hits (b).

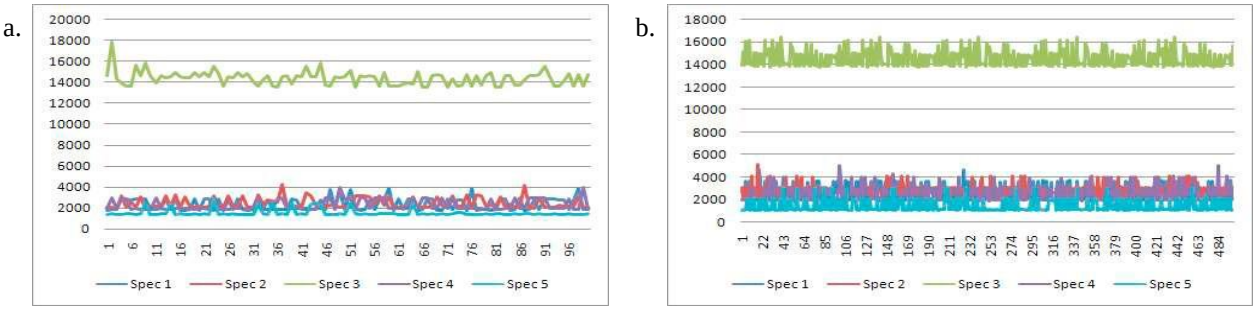

Fig 10. Scenario four test results for 100 hits (a); and 500 hits (b).

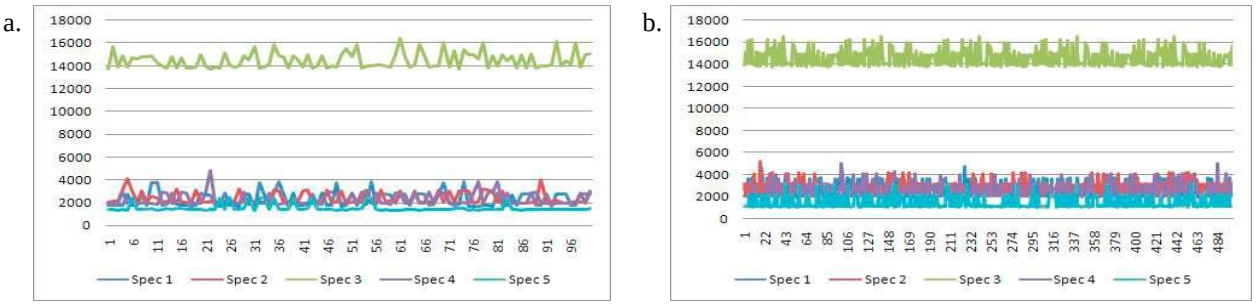

Fig 11. Scenario five test results for 100 hits (a); and 500 hits (b).

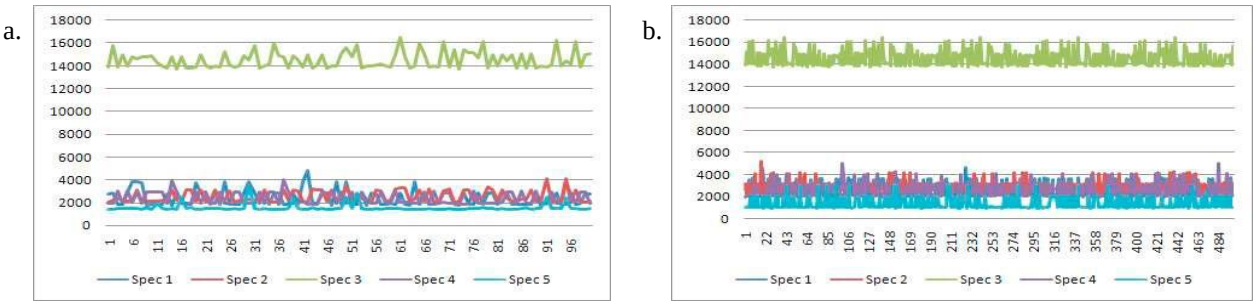

Fig 12. Scenario six test results for 100 hits (a); and 500 hits (b).

从图中列出的数据（如上面所示的线图所示），研究人员将获取主要数据，即： 每个方案与每个样本的平均加载速度，每个方案与每个样本的最低加载速度，每个方案与每个样本的最高加载速度，等于每个方案与每个样本中最频繁出现的加载速度以及加载速度值 每个场景与每个标本的中间位置。 所有数据列于表4，表5和表6。

Table 4. 每个样本中方案1和2的访问速度结果。

Specimen	Load Speed (milisecond)				Specimen	Load Speed (milisecond)			
	Average	*Slow*	*Fast*	*Modus*		*Average*	*Slow*	*Fast*	*Modus*
1	2,851.27	5,526.00	1,750.00	2,603.00	1	2,887.16	5,560.00	1,760.00	2,817.00

Specimen		Load Speed (milisecond)				Specimen		Load Speed (milisecond)			
		Average	*Slow*	*Fast*	*Modus*			*Average*	*Slow*	*Fast*	*Modus*
Scenario 1 Hit 100	2	2,483.76	5,312.00	2,009.00	2,094.00	*Scenario 2 Hit 100*	2	2,447.87	5,292.00	1,960.00	2,083.00
	3	14,501.3	16,428.0	13,729.0	14,928.0		3	14,537.2	16,438.0	13,774.0	14,814.0
	4	2,189.44	3,869.00	1,827.00	1,871.00		4	2,153.55	3,846.00	1,785.00	1,810.00
	5	1,667.76	3,995.00	954.00	1,972.00		5	1,703.65	4,015.00	984.00	2,013.00
Scenario 1 Hit 500	1	2,308.89	3,907.00	1,711.00	1,789.00	*Scenario 2 Hit 500*	1	2,344.78	3,957.00	1,748.00	1,834.00
	2	3,098.85	5,638.00	2,183.00	3,041.00		2	3,062.96	5,603.00	2,133.00	2,938.00
	3	14,291.1	17,854.0	13,516.0	13,569.0		3	14,326.9	17,889.0	13,536.0	13,745.0
	4	2,823.50	4,526.00	1,831.00	2,930.00		4	2,787.61	4,486.00	1,774.00	2,871.00
	5	1,518.25	4,044.00	987.00	1,021.00		5	1,554.14	4,094.00	1,006.00	1,058.00

Table 5.
每个样本中方案3和方案4的访问速度结果.

Specimen		Load Speed (milisecond)				Specimen		Load Speed (milisecond)			
		Average	*Slow*	*Fast*	*Modus*			*Average*	*Slow*	*Fast*	*Modus*
Scenario 3 Hit 100	1	2,400.45	3,920.00	1,769.00	2,844.00	*Scenario 4 Hit 100*	1	2,328.50	3,853.00	1,735.00	1,796.00
	2	2,466.06	4,029.00	2,035.00	2,164.00		2	2,485.16	4,283.00	1,985.00	2,110.00
	3	14,141.41	16,692.0	13,536.0	13,817.0		3	14,417.9	17,854.0	13,564.0	14,656.0
	4	2,328.62	3,085.00	1,928.00	2,007.00		4	2,295.79	3,969.00	1,898.00	2,010.00
	5	1,587.41	2,522.00	1,421.00	1,468.00		5	1,566.00	2,873.00	1,398.00	1,443.00
Scenario 3 Hit 500	1	2,812.76	4,601.00	2,495.00	2,613.00	*Scenario 4 Hit 500*	1	2,816.69	4,668.00	2,492.00	2,657.00
	2	3,198.15	5,070.00	2,766.00	3,136.00		2	2,608.31	5,132.00	1,998.00	3,080.00
	3	14,501.30	16,428.0	13,729.0	14,928.0		3	14,537.2	16,448.0	13,764.0	16,154.0
	4	2,369.24	5,043.00	1,864.00	1,916.00		4	2,414.12	4,999.00	1,882.00	1,974.00
	5	1,370.43	4,025.00	987.00	1,019.00		5	1,408.47	3,100.00	1,004.00	1,048.00

Table 6.
每个样本中方案5和方案6的访问速度结果

Specimen		Load Speed (milisecond)				Specimen		Load Speed (milisecond)			
		Average	*Slow*	*Fast*	*Modus*			*Average*	*Slow*	*Fast*	*Modus*
Scenario 5 Hit 100	1	2,315.25	3,830.00	1,694.00	1,770.00	*Scenario 6 Hit 100*	1	2,432.19	4,868.00	1,752.00	1,844.00
	2	2,387.60	4,074.00	1,974.00	2,054.00		2	2,543.92	4,152.00	2,051.00	2,160.00
	3	14,463.10	16,388.0	13,686.0	14,764.0		3	14,539.2	16,470.0	13,751.0	13,869.0
	4	2,273.23	4,868.00	1,837.00	1,972.00		4	2,397.68	4,053.00	1,903.00	1,959.00
	5	1,537.26	2,501.00	1,336.00	1,399.00		5	1,586.87	3,489.00	1,438.00	1,470.00
Scenario 5 Hit 500	1	2,852.58	4,746.00	2,517.00	2,674.00	*Scenario 6 Hit 500*	1	2,819.11	4,686.00	2,486.00	2,613.00
	2	2,644.20	5,177.00	2,023.00	2,132.00		2	2,677.67	5,212.00	2,037.00	2,135.00
	3	14,573.08	16,498.0	13,794.0	14,106.0		3	14,539.6	16,488.0	13,739.0	13,981.0
	4	2,450.01	5,031.00	1,912.00	1,980.00		4	2,483.48	5,041.00	1,932.00	2,040.00
	5	1,444.36	3,139.00	1,014.00	1,088.00		5	1,410.89	3,113.00	963.00	1,058.00

4.2. 讨论结果

根据表4，表5和表6中的数据，将每个样本的数据访问速度与每种情况进行比较，请注意:

- 每个测试场景中的样本4和5的访问速度在两者中都非常重要。

每个循环访问中的平均速度以及某些循环访问中可以达到的最高速度。

- 与每个测试场景中的样本4和5相比，某些测试场景中的样本1和2具有足够高的速度，但不够高，而对于100个所有测试场景，样本1和2足够高。
- 每个测试场景的三个样本在所有测试场景中都有较低的速度趋势。
- 对于最高的平均速度，可以在五个标本中的三个命中500的情况下看到，速度为1,370.45毫秒。
- 在第五个标本中以954毫秒的速度一次命中100次的情况下，可以看到最高的访问速度。
- 对于最低的平均访问速度，可以在场景5中以14573.08毫秒的速度命中样本3达到500次。
- 而对于具有第六个比较讨论数据的最低访问速度，场景五以13794毫秒的速度在标本3上命中500。

从表4，表5和表6得出的临时结论中，研究人员可以得出以下结论：第5个样品以最高速度，然后是第4个样品，然后是第1个和第2个样品，然后第3个样品处于进入速度的最后位置。

5. 结论

通过研究针对几种桥接框架的Spring框架的性能的研究，仅关注已完成的数据加载URL，可以知道一些结果，即：

- Spring Framework 4.0.1和MyBatis Framework 3.4.2之间的合并产生了非常积极的性能，然后将其添加到缓存引擎后，性能会提高。
- Spring Framework 4.0.1与Hibernate Framework 4.3的混合使用产生了积极的性能，但是当添加到第二级缓存时，将方向更改为负面。
- Spring Framework 4.0.1与Java数据库连接（JDBC）的混合通常比较适中，测试结果在中间，不是负面的也不是正面的。
- 因此，研究人员倾向于选择Spring框架和MyBatis框架的组合作为高性能的Web服务生成器，特别是在添加了缓存功能的情况下。

致谢

这项研究部分由Langlangbuana大学工程学院信息学系资助。

参考文献

[1] Begin, Clinton. (2007) *Ibatis in Action*, Manning Publication, New York, USA.
[2] Jogiyanto, H.M. (2008) *Analisis dan Perancangan Sistem: Pendekatan Terstruktur Teori dan Praktek Aplikasi Bisnis* [Title in English: *System Analysis and Design: Structured Approach including Theory and its Application in Busines*], Andi Offset., Yogyakarta., Indonesia.
[3] Johnson, Rod. (2005) *Professional Java Development with the Spring Framework*, Wiley Publishing., Indianapolis, USA.
[4] Kalelkar, Medha., Churi, Prathamesh., Kalelkar, Deepa. (2014) Implementation of Model-View-Controller Architecture Pattern for Business Intelligence Architecture., International Journal of Computer Application., New York, USA.
[5] Kendal, Simon. (2009) *Object Oriented Programming using Java*., Ventus Publishing Aps, Frederiksberg, Denmark.
[6] Sugiyono. (2012) *Metode Penelitian Kuantitatif Kualitatif dan R&D* [Title in English: *Qualitative and Quantitative Research Method and R&D*], Alfabeta, Bandung. Indonesia.
[7] Zimanyi, Esteban. (2014) *Introduction to Java Hibernate*, Universite libre de Bruxelles, Bruxelles, Belgia.
[8] Roger, S. Pressman. (2012) Rekayasa Perangkat Lunak [Title in English: *Software Engineering*], 7th Edition : Book 1, Andi Offset, Yogyakarta., Indonesia.
[9] Nacheva, Radka. (2017) *Prototyping Approach in User Interface Development, Conference on Innovative Teaching Methods*, University of Economics Varna, Bulgaria.