SO SÁNH HIỆU NĂNG CỦA CÁC FRAMEWORK TẬP TRUNG XỬ LÝ PHÍA MÁY CHỦ TRÊN NỀN TẢNG WEB VỚI CÔNG NGHỆ JAVA VÀ ỨNG DỤNG TẠI HỌC VIỆN NÔNG NGHIỆP VIỆT NAM

Trần Trung Hiếu*, Phạm Quang Dũng, Đỗ Thị Nhâm

Khoa Công nghệ thông tin, Học viện Nông nghiệp Việt Nam

**Tác giả liên hệ: tthieu@vnua.edu.vn*

Ngày nhận bài: 20.07.2020 Ngày chấp nhận đăng: 23.12.2020

TÓM TẮT

Trong nghiên cứu này, chúng tôi thực hiện so sánh hiệu năng của hai framework tiêu biểu trong nhóm các framework tập trung xử lý phía máy chủ trên nền tảng web với công nghệ Java là ZK và JSF hướng đến một lựa chọn sử dụng. Bằng phương pháp kiểm thử hiệu năng dựa trên các tiêu chí thời gian phản hồi và dung lượng bộ nhớ sử dụng, nghiên cứu cách mô phỏng, cấu hình thông số trên các công cụ Jmeter, VisualVM và máy ảo Java, các kết quả đưa ra phản ánh khách quan hiệu năng của hai ứng dụng được xây dựng bởi mỗi framework. Qua kết quả kiểm thử hiệu năng, chúng tôi đã chọn ZK là framework có hiệu năng tốt hơn JSF. ZK framework đã được chúng tôi áp dụng vào thực tiễn, xây dựng các ứng dụng đã được đưa vào sử dụng ở Học viện Nông nghiệp Việt Nam.

Từ khóa: ZK, JSF, PrimeFaces, framework, server-centric, hiệu năng.

Comparing Performances of Server-Centric Java Web Frameworks and Applying at Vnua

ABSTRACT

In this study, we compared the performances of two typical frameworks in the server-centric java web framework group, ZK and JSF, towards a usage option. Using a performance test method based on the response time and memory usage criteria, studying simulations, parameter configurations on Jmeter, VisualVM and Java virtual machines were made. The results revealed objective reflection on the performance of the two applications built by each framework. Through the performance test results, we chose ZK, which had a better performance framework than JSF. ZK framework was the applied in practice, building applications that have been put into use at Vietnam National University of Agriculture.

Keywords: ZK, JSF, PrimeFaces, framework, server-centric, Jmeter, VisualVM, performance.

1. ĐẶT VẤN ĐỀ

Trong 20 năm trở lại đây, ngôn ngữ lập trình Java luôn là lựa chọn hàng đầu để xây dựng các phần mềm ứng dụng, trong nhiều năm Java thường xếp thứ nhất hoặc nhì trong bảng xếp hạng các ngôn ngữ lập trình được sử dụng nhiều nhất (Bảng 1).

Các framework lập trình ứng dụng Web với công nghệ Java được chia thành 2 nhóm, nhóm client-centric phân tán nhiều công việc xử lý về phía máy khách, nhóm server-centric tập trung đa phần xử lý ở phía máy chủ (1&1 IONOS Inc, 2017). Nhóm các client-centric framework đòi hỏi lập trình viên cần am hiểu về các ngôn ngữ HTML, Javascript. Do đặc thù của những ngôn ngữ này, chương trình thường phức tạp hơn và tốn nhiều thời gian công sức để phát triển hơn. Nhóm các server-centric framework đã xây dựng sẵn nhiều thư viện để đơn giản hóa việc lập trình của người sử dụng nên thường dễ phát triển hơn, thời gian xây dựng ngắn hơn và do tập trung phần lớn công việc xử lý ở phía server nên tính bảo mật cũng cao hơn, đây là một tiêu

chí quan trọng trong lựa chọn của nhiều doanh nghiệp. Tốc độ xử lý của các client-centric framework thường nhanh hơn các server-centric framework, tuy nhiên cân nhắc trên nhiều yếu tố, trong nhiều trường hợp, server-centric framework là một lựa chọn tốt.

Trong một số server-centric web framework xây dựng trên nền tảng ngôn ngữ lập trình Java như JavaServer Faces (JSF), Wicket, Tapestry, Vaadin, ZK, thì JSF được sử dụng nhiều nhất (Perforce Software Inc, 2020). JSF ra đời năm 2001, là một framework mã nguồn mở được viết bởi Sun Microsystems. JSF thường phải sử dụng kèm với một thư viện hỗ trợ tạo giao diện người dùng như PrimeFaces (PrimeTek Informatics, 2020) hay RichFaces (RedHat, 2020). Hiện nay, JSF được hỗ trợ phát triển bởi Oracle, tập đoàn tiếp quản công nghệ Java từ Sun Microsystems, JSF đã được Oracle đưa thành chuẩn Java cho các ứng dụng đồ họa trên nền web (Scholtz & cs., 2018), JSF thường được dùng kết hợp với PrimeFaces nhiều nhất, giá của thư viện GUI (Graphic User Interface) này rẻ bất ngờ với chỉ 19-79 đối với các layout riêng lẻ, hay chưa đến 200$ cho một phiên bản nhiều người dùng (PrimeTek Informatics, 2020), đây có lẽ là hai lý do chính JSF được sử dụng nhiều hơn các framework khác, dù nó có một số nhược điểm. ZK được viết bởi Potix Đài Loan năm 2005 (Potix, 2020), ZK bao gồm hai phiên bản, phiên bản thương mại và phiên bản mã nguồn mở dành miễn phí cho cộng đồng. Dù ra đời sau, ZK đã nhanh chóng phát triển và chiếm được sự tin tưởng của nhiều công ty, tập đoàn lớn trên thế giới như Samsung, Sony, Toyota, HTC, Airbus, Barclays, Ebay, Bank of America, Deutsche Bank, US Department of Defense… và chính Oracle cũng sử dụng ZK. Triết lý của ZK là "Ajax without Javascript", nó cho phép người dùng xây dựng ứng dụng web mà không cần biết bất kỳ kiến thức nào về Ajax và Javascript. ZK hỗ trợ số lượng lớn các thành phần giao diện đã được xây dựng sẵn với trên 200 thành phần, là framework đầu tiên hỗ trợ bảng tính (spreadsheet), mô hình MVVM (Model-View-ViewModel) trên nền web. Theo phản hồi từ phía người sử dụng, thời gian xây dựng ứng dụng với ZK giảm từ 4 đến 16 lần. ZK bảo vệ các ứng dụng chống lại các hình thức tấn công XSS, DoS và CSRF, ZK tăng cường hơn nữa xác thực và ủy quyền với các khuôn khổ bảo mật của bên thứ ba như Spring Security, hoàn toàn cung cấp bảo vệ từ cấp độ trang đến các sự kiện Ajax, đây cũng là lý do quan trọng để các doanh nghiệp lớn lựa chọn ZK. Ngoài ra, ZK cho phép tích hợp nhiều công nghệ khác như JSP, Struts, Spring, EJB, Hibernate, CDI, JDBC, Bootstrap (Potix, 2020)… với tài liệu hướng dẫn chi tiết, đây là một yếu tố rất quan trọng khi các ứng dụng lớn cần kết hợp nhiều công nghệ khác nhau.

Về mặt tính năng, ZK có nhiều ưu điểm vượt trội so với JSF. Tuy nhiên, các tính năng, chức năng của một hệ thống phần mềm không phải mối quan tâm duy nhất, hiệu năng của một phần mềm cũng là một yếu tố đáng chú ý. Hiệu năng bao gồm các yếu tố như thời gian phản hồi (response time), độ tin cậy (reliability), mức độ sử dụng tài nguyên (resource usage) và khả năng mở rộng (scalability) (Bathia & cs., 2018), các yếu tố này ảnh hưởng không nhỏ đến chất lượng dịch vụ cung cấp cho người dùng, cũng như sự tiêu tốn tài nguyên phía nhà cung cấp phải đáp ứng. Nghiên cứu này nhằm so sánh hiệu năng của hai framework, và thực nghiệm với hai framework tiêu biểu của nhóm các server-centric java web framework là JSF và ZK, từ đó lựa chọn ra một framework tốt hơn cho xây dựng ứng dụng.

2. PHƯƠNG PHÁP NGHIÊN CỨU

Có nhiều loại kiểm thử hiệu năng bao gồm: load test (kiểm thử khả năng tải), stress test (kiểm thử xem hệ thống hoạt động như thế nào khi quá tải và cách hệ thống phục hồi khi xảy ra lỗi), capacity test (kiểm thử lượng giao dịch trên một đơn vị thời gian), endurance test (kiểm thử lượng tải ổn định trong một khoảng thời gian dài), spike test (kiểm thử phản ứng của phần mềm trước các thay đổi lớn hoặc đột ngột khi tải), volume test (kiểm thử hiệu suất của hệ thống ứng với các khối lượng cơ sở dữ liệu khác nhau), s*calability test (kiểm thử khả năng mở rộng tải của ứng dụng), reliability test (kiểm thử độ tin cậy của hệ thống, hay khả năng thực hiện một hoạt động không có lỗi trong một khoảng thời gian nhất định)* (SoftwareTestingHelp, 2020).

Bảng 1. Thứ hạng ngôn ngữ lập trình Java qua 20 năm theo Tiobe

Programming Language	2020	2015	2010	2005	2000
Java	1	2	1	2	3

"Load test" là một quá trình thêm nhu cầu vào một hệ thống hoặc thiết bị và đo lường phản ứng của nó. Load testing được thực hiện để xác định ứng xử của hệ thống trong các điều kiện tải bình thường và cao hơn điều kiện tải dự kiến (Try QA, 2020). "Load test" có thể được sử dụng để so sánh hiệu năng của các ứng dụng. Ví dụ, m*ột nghiên cứu so sánh hiệu năng ứng dụng web trên hai nền tảng.NET và Java EE (*Enterprise Edition) sử dụng phương pháp "Load test" với công cụ Load Runner và Webking để đo thời gian phản hồi và dung lượng bộ nhớ sử dụng ứng với các trường hợp giả lập số người dùng khác nhau trên cùng một ứng dụng (Hamed & cs., 2009). Tại website trang chủ của ZK framework, nghiên cứu của tác giả James Chu so sánh hiệu năng của phiên bản ZK 7, ZK 8 được thực hiện trên một máy tính cá nhân thông thường, sử dụng công cụ Apache Jmeter để giả lập người dùng và kiểm thử thời gian phản hồi, công cụ Visual VM để kiểm thử dung lượng bộ nhớ sử dụng ứng với các trường hợp giả lập số người dùng tăng dần từ 100, 200 tới 1.500 trong thời gian một giây, ứng dụng thử nghiệm có giao diện khá nặng với khoảng 900 khối div và label (Potix, 2020).

Để so sánh hiệu năng của hai framework, trong bài báo này chúng tôi chọn loại kiểm thử "load test" để so sánh thời gian phản hồi (response time), dung lượng bộ nhớ sử dụng (memory consumption) và phần trăm sử dụng CPU tương ứng. Các bước kiểm thử bao gồm: xác định môi trường kiểm thử, thiết kế trường hợp kiểm thử (testcase), cài đặt môi trường kiểm thử và thực hiện kiểm thử.

2.1. Xác định môi trường kiểm thử

Về phần cứng, chúng tôi sử dụng máy cục bộ (localhost) đóng vai trò làm webserver để chạy hai ứng dụng ứng với hai framework có cấu hình:

CPU: Intel(R) Core(TM) i5-3210M Processor @ 2.50GHz

Bộ nhớ: 8GB

Về phần mềm, chúng tôi sử dụng các phần mềm sau:

ZK 9.0.0 CE: phiên bản ZK framework miễn phí dành cho cộng đồng

JSF 2.1: sử dụng kết hợp với PrimeFaces

PrimeFaces 8.0: sử dụng phiên bản miễn phí dành cho cộng đồng

JDK 1.8.0_241: Bộ công cụ phát triển Java, bao gồm máy ảo Java

Apache Tomcat 9: Webserver

Eclipse IDE 2020-03: Môi trường phát triển tích hợp cho ứng dụng Java EE

Apache Jmeter 5.3 (The Apache Software Foundation, 2020): Công cụ kiểm thử hiệu năng ứng dụng

Visual VM 1.4.2 (The Apache Software Foundation, 2020): Công cụ kiểm thử bộ nhớ sử dụng cho các ứng dụng Java

Với ZK, vì không có điều kiện sử dụng phiên bản mất phí nên chúng tôi sử dụng phiên bản miễn phí dành cho cộng đồng. Phiên bản ZK, JSF, PrimeFaces được chọn là mới nhất tại thời điểm tiến hành kiểm thử, các phiên bản phần mềm còn lại có thể tùy chọn, sao cho đảm bảo sự tương thích là được.

Về các thông số cấu hình, để đảm bảo việc so sánh được khách quan, chúng tôi cần cấu hình một số thông số trong webserver sao cho cơ chế quản lý của webserver không gây ảnh hưởng đến việc so sánh. Các thông số cấu hình Tomcat webserver bao gồm:

Session timeout mặc định: Thời gian kiểm thử ứng dụng là ngắn nên không cần cấu hình đặc biệt gì cho tham số này

- Xms 2048MB: kích thước bộ nhớ heap khởi tạo khi máy ảo Java hoạt động

- Xmx 2048MB: kích thước bộ nhớ heap tối đa

- XX: MetaspaceSize 1024MB: kích thước bộ nhớ Metaspace khởi tạo

- XX: MaxMetaspaceSize 1024MB: kích thước bộ nhớ Metaspace tối đa

maxThreads: 2000: số yêu cầu tối đa có thể được xử lý đồng thời bởi webserver, mặc định giá trị này trong tomcat là 200 (The Apache Software Foundation, 2020)

acceptCount: 1024: Độ dài hàng đợi tối đa cho các yêu cầu kết nối đến webserver. Mọi yêu cầu nhận được khi hàng đợi đầy sẽ bị từ chối. Giá trị mặc định là 100 (The Apache Software Foundation, 2020). Số lượng người dùng tối đa trong thử nghiệm là 1000 nên giá trị này được thiết lập là 1024.

Bộ nhớ heap chứa các đối tượng (Object) Java, các biến tĩnh, các biến toàn cục. Bộ nhớ heap được dọn dẹp tự động bởi bộ dọn rác GC (garbage collector), mặc định kích thước bộ nhớ heap tối đa là 64MB (Oracle, 2020). Mỗi framework sử dụng bộ nhớ Heap khác nhau, để việc dọn dẹp bộ nhớ bởi bộ GC mà ta không đoán định được không ảnh hưởng chương trình, và phòng khi kiểm thử số lượng người dùng truy cập lớn có thể thiếu bộ nhớ, chúng tôi thiết lập kích thước bộ nhớ Heap khi khởi tạo cũng như tối đa bằng nhau, đều là 2048MB.

Trước phiên bản Java 8, bộ nhớ PermGen nằm trong Heap được sử dụng khi ứng dụng cần tải số lượng lớn các lớp (class), do bị hạn chế bởi kích thước bộ nhớ Heap, nên ứng dụng dễ có khả năng bị lỗi tràn bộ nhớ (java.lang. OutOfMemoryError: PermGen space). Từ phiên bản Java 8, bộ nhớ PermGen được loại bỏ, thay thế bởi bộ nhớ Metaspace (Oracle, 2020). Metaspace không phải một phần của bộ nhớ Heap như PermGen, kích thước của nó không giới hạn, tùy thuộc vào bộ nhớ được hệ điều hành cấp cho máy ảo Java. Bộ nhớ Metaspace tuy không giới hạn nhưng ta nên đặt giới hạn cho nó để tránh việc rò rỉ bộ nhớ (memory leak).

2.2. Thiết kế kịch bản kiểm thử

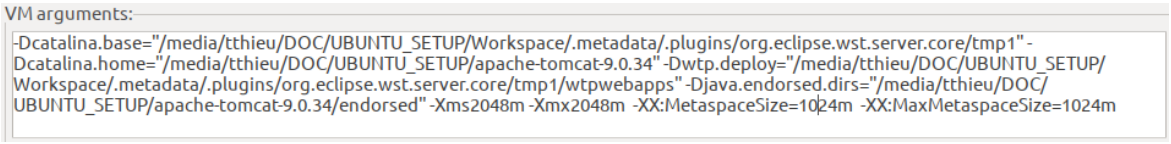
Để so sánh hiệu năng của hai framework, chúng tôi kiểm tra khả năng của hai ứng dụng được xây dựng từ hai framework, thực hiện tải trang web có chứa một bảng dữ liệu với 9 cột và 2.000 dòng, mô phỏng với lần lượt 100, 200,… 1.000 người dùng đồng thời truy cập trong 1s và ghi nhận thời gian phản hồi, dung lượng bộ nhớ sử dụng.

Bảng 2. Mẫu testcase kiểm thử so sánh hai framework

Mã testcase	Mô tả testcase	Dữ liệu kiểm thử	Các bước thực hiện	Thời gian phản hồi		Bộ nhớ sử dụng		% CPU sử dụng	
				JSF	ZK	JSF	ZK	JSF	ZK
XX	Kiểm thử khả năng tải N user dùng đồng thời truy cập trong 1s	Trang web với bảng dữ liệu 9 cột, 2.000 dòng	1. Mô phỏng N user đồng thời truy cập trong 1s trên Jmeter 2. Đồng thời theo dõi bộ nhớ sử dụng trên VisualVM 3. Xuất tệp csv kết quả trên hai công cụ khi kết thúc	?	?	?	?	?	?

ID	First Name	Last Name	SurName	Year	Country	Address	Class	Faculty
0	Thien	G	C	1992	Viet Nam	Quang Ninh	B	CNTT
1	Nhan	H	G	2000	Viet Nam	Hai Phong	F	NH
2	Tam	I	B	1999	Viet Nam	Thai Binh	X	KT-QTKD

Hình 1. Mẫu dữ liệu kiểm thử minh họa cho 2.000 bản ghi



Hình 2. Thiết lập thông số cho máy ảo Java trên Eclipse

2.3. Thiết lập môi trường kiểm thử

Môi trường kiểm thử được thực hiện trên máy cục bộ với cấu hình đã khai báo. Các phần mềm được cài đặt theo thứ tự: JDK, Jmeter, VisualVM, Eclipse. Webserver Tomcat được cài đặt trong môi trường phát triển tích hợp Eclipse.

Các thông số cấu hình cho máy ảo Java được thực hiện tại chế độ chạy cấu hình ứng dụng (Run configuration) trên Eclipse (Hình 2).

Các thông số cấu hình cho webserver Tomcat gồm maxThreads và acceptCount được cấu hình trong file server.xml nằm trong thư mục cài đặt Tomcat:

<Connector port="8080" protocol= "HTTP/1.1" connectionTimeout="20000" redirectPort="8443" acceptCount="1024" maxThreads="2000"/>

Hai project ứng dụng được xây dựng trên Eclipse và thực hiện chạy trên webserver Tomcat. Giá trị các trường dữ liệu được sinh ngẫu nhiên từ một mảng các giá trị tương ứng. Dữ liệu cung cấp cho tầng xử lý hiển thị của hai framework được thiết kế giống nhau và được sinh động, mỗi bản ghi dữ liệu ứng với một đối tượng được tạo ra trong bộ nhớ nên nó cũng có tác dụng làm tăng yêu cầu kiểm thử khả năng tải của webserver với ứng dụng trên framework tương ứng.

Để giả lập truy cập đồng thời của nhiều người sử dụng vào một ứng dụng chúng tôi sử dụng công cụ Jmeter. Jmeter thực hiện giả lập một nhóm người dùng gửi các yêu cầu tới một máy chủ, nhận và xử lý các phản hồi từ máy chủ và cung cấp các kết quả báo cáo hiệu suất dưới dạng biểu đồ trực quan, dễ hiểu (Hình 3).

Các tham số cần thiết lập trên Jmeter:

- Thread group:

Number of Thread (users): N

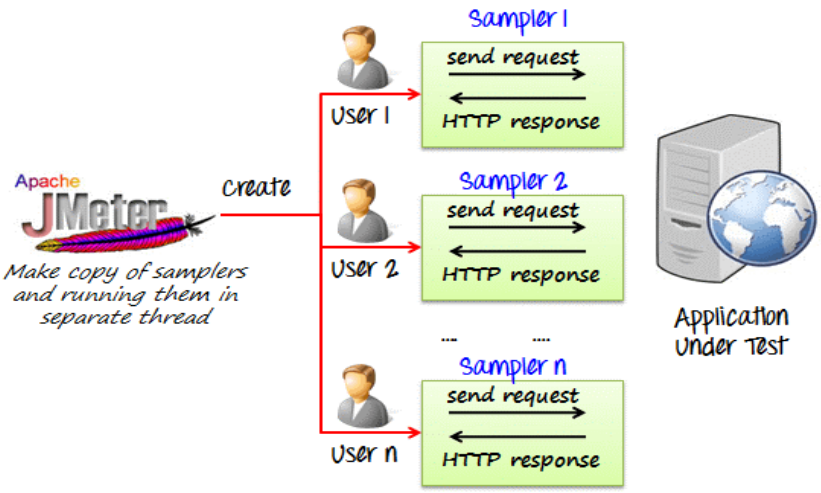
Số người dùng đồng thời truy cập hệ thống, N được thay đổi trong mỗi lần kiểm thử

Ramp-up period (seconds): 1

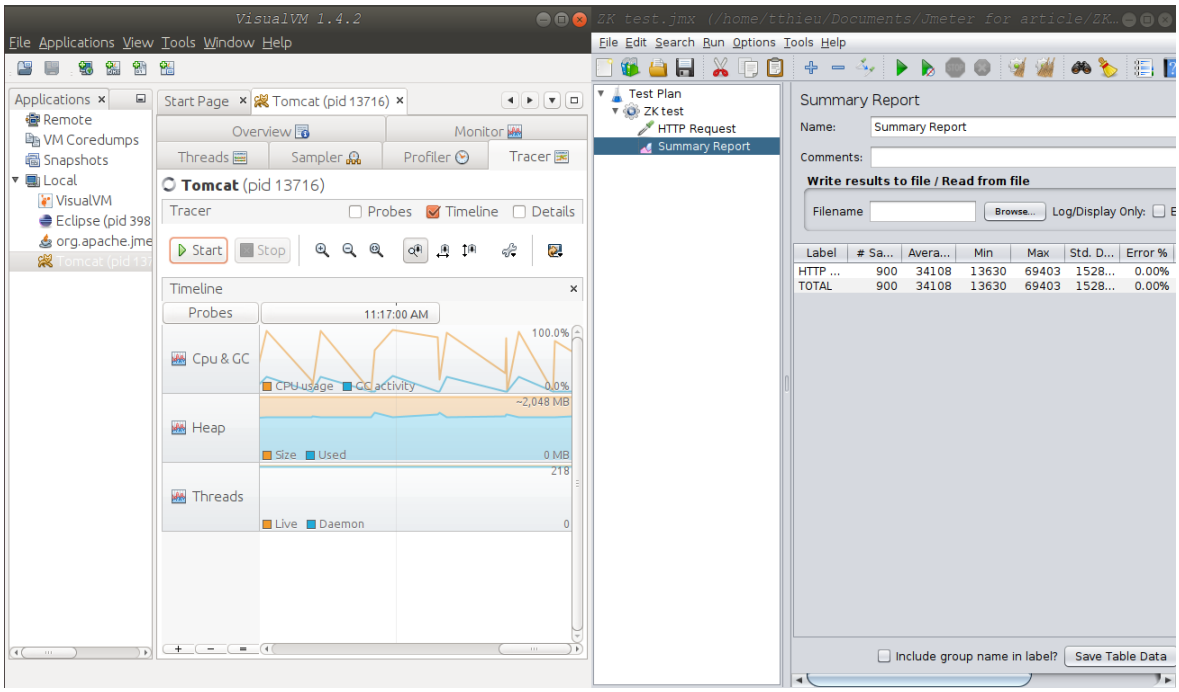
Khoảng thời gian mà N người dùng đồng thời truy cập

Loop count: 1

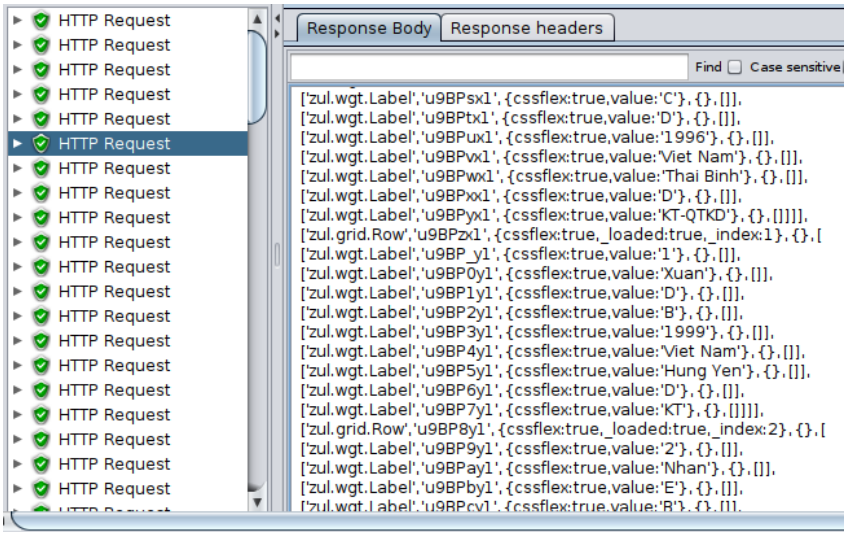
Số lần kiểm thử lặp lại



Hình 3. Cơ chế giải lập của Jmeter (Guru99, 2020)



Hình 4. Cách bố trí cửa sổ hai ứng dụng kiểm thử để đảm bảo đồng bộ dữ liệu



Hình 5. Dữ liệu phản hồi từ các request trên Jmeter gửi tới ứng dụng ZK

- Http request:

Protocol: http

Giao thức truy cập ứng dụng web

Server name or IP: localhost

Địa chỉ server ứng dụng

Port Number: 8080

Số cổng lắng nghe yêu cầu của webserver tomcat

Request: GET

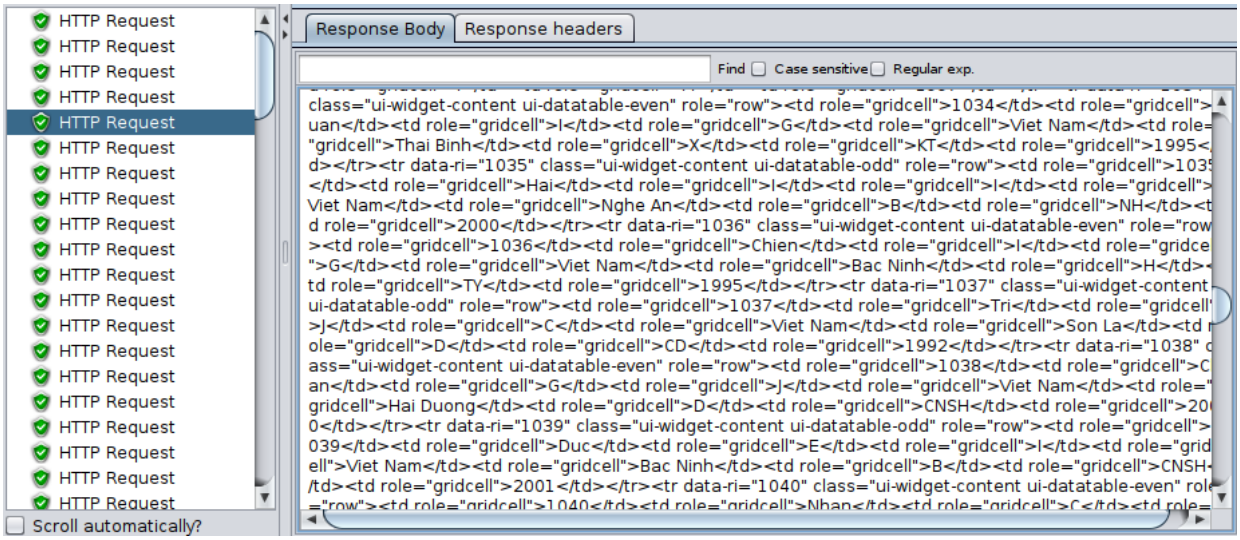
Phương thức gửi request tới webserver

Path: http://localhost:8080/TestZKforArticle/

Đường dẫn truy cập ứng dụng, thay đường dẫn tương ứng với ứng dụng JSF

- Summary Report: Báo cáo tổng hợp cần thêm vào trên Thread group

- View results tree: Cây theo dõi các request được gửi và dữ liệu phản hồi từ server cần thêm vào trên Thread group



Hình 6. Dữ liệu phản hồi từ các request trên Jmeter gửi tới ứng dụng JSF

Trên công cụ VisualVM cần cài đặt thêm plugin Tracer-Monitor để có thể theo dõi và xuất dữ liệu ra dưới dạng tệp csv.

2.4. Thực hiện kiểm thử

Để tiến hành kiểm thử cho mỗi trường hợp N người sử dụng truy cập vào ứng dụng, ta cần bật webserver, bố trí hai ứng dụng Jmeter và VisualVM song song trên cửa sổ màn hình. Đồng thời khởi động kiểm thử mô phỏng trên Jmeter, và kiểm thử bộ nhớ sử dụng trên VisualVM, khi việc chạy trên Jmeter kết thúc, cũng đồng thời nhấn kết thúc chương trình chạy trên VisualVM. Việc xuất kết quả ra tệp csv trên hai ứng dụng diễn ra sau đó. Sau mỗi lần như vậy ta thu được 2 tệp csv kết quả. Để kiểm thử với mỗi giá trị số người sử dụng N khác nhau, cần thực hiện khởi động lại webserver để đảm bảo tài nguyên sử dụng cho lần kiểm thử trước được giải phóng hoàn toàn

3. KẾT QUẢ VÀ THẢO LUẬN

Về tính đáng tin cậy của việc giả lập trên Jmeter, kiểm tra nội dung request gửi trên "View Result Tree" trong Jmeter, ta thấy nội dung dữ liệu phản hồi từ webserver phản ánh đúng mã nguồn trang web tương ứng trên trình duyệt với bảng dữ liệu. Điều này củng cố thêm tính tin cậy của việc giả lập trên Jmeter. Có những framework như Vaadin ta rất khó thực hiện kiểm thử trên Jmeter bởi dữ liệu phản hồi từ webserver có chứa mã động javascript, Jmeter chỉ có thể tải một phần dữ liệu mong muốn dẫn đến kết quả kiểm thử không chính xác.

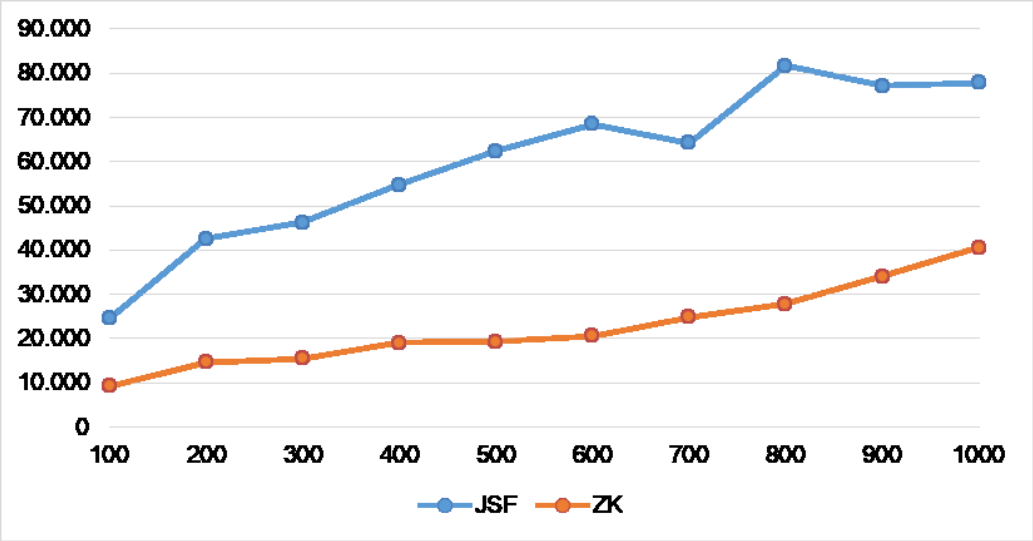
Thực hiện kiểm thử theo mẫu testcase đã đề ra và phương pháp đưa ra ở mục 2.4, ta thu được bảng số liệu (Bảng 3), từ đó ta vẽ được các biểu đồ so sánh thời gian phản hồi (Hình 7), phần trăm CPU sử dụng (Hình 8) và bộ nhớ sử dụng (Hình 9).

Qua biểu đồ so sánh thời gian phản hồi, ta thấy cả hai framework đều có thời gian phản hồi tăng dần khi số người dùng truy cập tăng dần. Thời gian phản hồi của ZK framework là nhanh hơn trong mọi trường hợp kiểm thử. Thời gian phản hồi trung bình với JSF framework là 59995,4 (ms), với ZK framework là 22607,6 (ms), tính trung bình thời gian phản hồi của ZK nhanh hơn JSF 2,65 lần.

Biểu đồ so sánh phần trăm sử dụng CPU của máy ảo Java cho thấy, JSF sử dụng CPU nhiều hơn và ít biến động hơn, trung bình JSF sử dụng là 87,9%. ZK sử dụng CPU ít hơn JSF, khi số người dùng truy cập tăng (từ 800 đến 1.000), ZK cho thấy xu hướng giảm phần trăm CPU sử dụng, đây là một điểm cộng của ZK. Trung bình ZK sử dụng 72,5% CPU.

Bảng 3. Bảng kết quả kiểm thử

Mã testcase	Mô tả testcase	Thời gian phản hồi trung bình(ms)		Bộ nhớ sử dụng (Byte)		CPU (%)	
		JSF	ZK	JSF	ZK	JSF	ZK
01	100 users/1s	24614	9275	337379315,2	448324734,77	80,38	61,35
02	200 users/1s	42534	14733	385484710,22	586512438,59	85,97	77,28
03	300 users/1s	46288	15550	412876526,96	722201673,33	86,86	85,04
04	400 users/1s	54741	19127	400760045,61	803781468,31	89,32	78,6
05	500 users/1s	62395	19286	396215852,92	828885223,38	91,13	77,35
06	600 users/1s	68553	20607	411764396,61	917078304,94	88,49	73,01
07	700 users/1s	64221	24934	396889074,54	959050239,79	88,95	76,42
08	800 users/1s	81624	27910	389539320,61	1039310375,83	89,71	78,9
09	900 users/1s	77185	34108	424241753,82	1195223791,89	89,54	67,03
10	1000 users/1s	77799	40546	438933267,6	1391515294,33	89,37	50,2

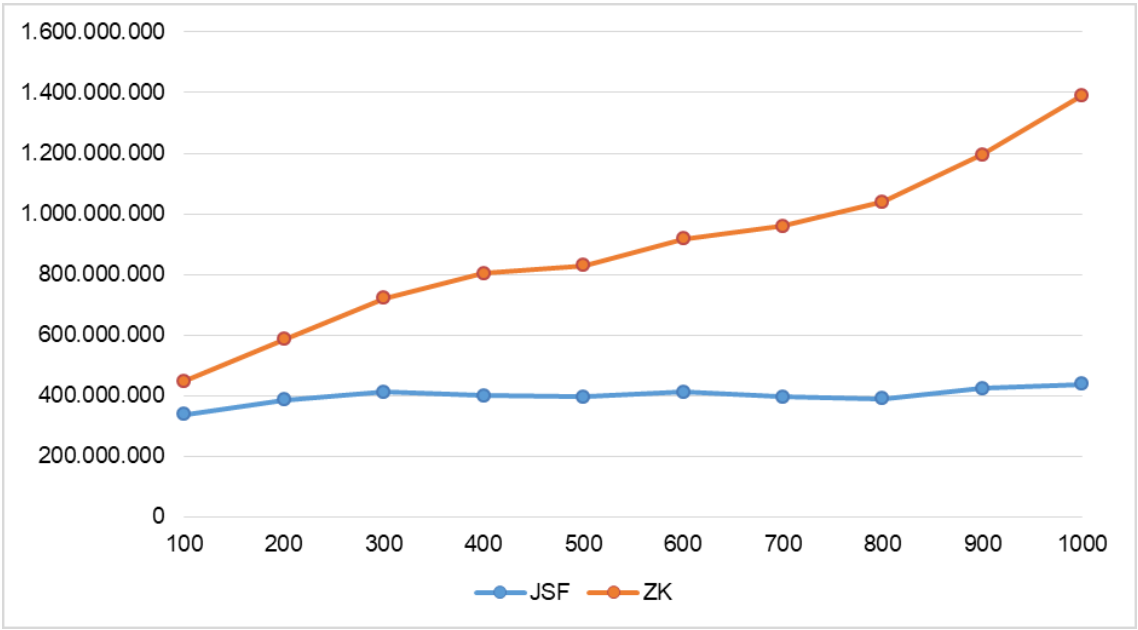


Hình 7. Biểu đồ so sánh thời gian phản hồi trung bình

Biểu đồ so sánh dung lượng bộ nhớ sử dụng cho thấy, JSF sử dụng ít bộ nhớ hơn ZK. Trong khoảng từ 100 đến 1.000 người sử dụng truy cập, bộ nhớ sử dụng của JSF là ít biến động, ZK sử dụng nhiều bộ nhớ hơn và có xu hướng tăng dần khi số lượng người dùng tăng. Trung bình JSF sử dụng 380,9 (MB), trung bình ZK sử dụng 848 (MB), trung bình bộ nhớ ZK sử dụng nhiều gấp 2,2 lần JSF.

Trong các biểu đồ kết quả chúng tôi thu được, đồ thị biểu diễn số liệu ứng với hai framework là tách biệt hoàn toàn, vì vậy trong nhận xét đánh giá, chúng tôi thấy cũng không cần thiết phải sử dụng thêm các mô hình kiểm nghiệm. Các thông số kết quả mà chúng tôi thu được trong phép kiểm thử khả năng tải trên môi trường máy cục bộ, như thời gian phản hồi, chênh lệch nhiều so với khoảng chấp nhận được trên môi trường triển khai thực, đó là do đặc thù của phép kiểm thử truy cập đồng thời trong thời gian ngắn, lượng dữ liệu cần tải lớn, việc khởi động lại webserver khiến một số đối tượng chức năng phải khởi tạo lại và năng lực hạn chế của máy cá nhân được dùng làm webserver. Tuy nhiên, các thông số đó là có giá trị giúp chúng tôi đạt được mục đích so sánh hiệu năng của hai framework.

Hình 8. Biểu đồ so sánh phần trăm CPU sử dụng



Hình 9. Biểu đồ so sánh dung lượng bộ nhớ sử dụng

4. ỨNG DỤNG

Sau khi tiến hành khảo sát, so sánh các framework của Java, chúng tôi quyết định sử dụng ZK framework để xây dựng website đặt cơm tại nhà hàng H+ Green VNUA của Học viện Nông nghiệp Việt Nam. Website nằm trong chương trình của Học viện hỗ trợ một phần tiền ăn trưa cho cán bộ Học viện, phối hợp với nhà hàng H+ Green Vnua cho phép các cán bộ học viện có thể đăng ký suất ăn trưa trực tuyến, quét thẻ xác nhận dùng cơm và thống kê, xuất báo cáo danh sách các cán bộ đã dùng cơm theo một khoảng thời gian nhất định.

Nhờ có những ưu điểm của ZK framework, chúng tôi có thể triển khai xây dựng website trong thời gian ngắn, website có giao diện đẹp, tối ưu hóa trải nghiệm người dùng nhờ các thành phần giao diện hỗ trợ mạnh bởi ZK framework.

Hình 10. Hình ảnh giao diện website đặt cơm vnua

Website phục vụ cho hơn 10.00 cán bộ của Học viện đăng ký ăn trưa và quét thẻ xác nhận hàng ngày, việc xuất báo cáo theo tháng với file excel cỡ lớn chứa từ 3.000 đến 4.500 bản ghi dữ liệu. Website vẫn hoạt động tốt kể từ khi triển khai tháng 9/2019 đến nay (tháng 6/2020). Trong thời gian xây dựng website, chúng tôi gặp rất ít lỗi phải sửa và trong thời gian từ khi triển khai, cũng có rất ít lỗi phải bảo trì, điều này có được cũng nhờ một phần ở tính ưu việt của công nghệ ZK framework. Website có thể truy cập vào từ địa chỉ: http://datcom.vnua.edu.vn/.

Hiện chúng tôi đang tiếp tục vận dụng công nghệ ZK framework vào một đề tài cấp Học viện để phát triển website cho khoa Công nghệ thông tin. Đề tài sẽ hoàn thành vào tháng 12/2020

5. KẾT LUẬN

Trong bài báo này, chúng tôi đã đưa ra được phương pháp so sánh hiệu năng của hai framework tiêu biểu trong nhóm các server-centric java web framework hướng đến một lựa chọn sử dụng. Các kết quả đưa ra phản ánh khách quan hiệu năng của hai ứng dụng được xây dựng bởi mỗi framework.

Qua kết quả kiểm thử hiệu năng, chúng tôi thấy được ZK framework tối ưu hơn về mặt thời gian phản hồi và phần trăm sử dụng CPU. Điểm trừ của ZK framework so với JSF framework là ở dung lượng bộ nhớ sử dụng, ZK framework sử dụng dung lượng bộ nhớ nhiều hơn nhưng điểm này hoàn toàn có thể chấp nhận được khi thực tế số lượng người dùng lớn cùng truy cập vào ứng dụng trong 1 giây là ít xảy ra và bộ nhớ của máy chủ thường lớn từ 32GB đến 2TB. Hơn nữa, nếu người dùng có điều kiện sử dụng các phiên bản thương mại của ZK framework, tốc độ phản hồi và dung lượng bộ nhớ sử dụng còn được cải thiện nhiều.

Cuối cùng, tổng hợp tất cả các so sánh ưu nhược điểm, chúng tôi quyết định lựa chọn ZK framework cho xây dựng ứng dụng của nhóm. Ứng dụng đã xây dựng thành công giảm thiểu công sức và thời gian, ứng dụng hiện đang hoạt động tốt, ổn định, đáp ứng nhanh, phục vụ cho công việc thực tế của cán bộ Học viện.

LỜI CẢM ƠN

Để hoàn thành bài báo trên, các tác giả xin bày tỏ lời cảm ơn đến Học viện Nông nghiệp Việt Nam đã phê duyệt đề tài cấp Học viện, mã số T2020-10-44.

TÀI LIỆU THAM KHẢO

1&1 IONOS Inc (2017). Web frameworks – overview and classification. Retrieved from https://www.ionos.com/digitalguide/websites/web-development/web-frameworks-an-overview on June 15, 2020.

Abdullah J.M., Mohammed M.A. & Muhammed D.A. (2015). Java 8 New Features Improvements and Complications. Int. J. of Multidisciplinary and Current research.

Bathia S. (2018). PSD: Professional Scrum Developer Question Bank and Reference Guide. Retrieved from https://books.google.com.vn/books?id=a4C3DwAAQBAJ&printsec=copyright&redir_esc=y#v=onepage&q&f=false on June 17, 2020.

Ciugudean M. & Gorgan D. (2016). Methodology for Identification and Evaluation of Web Application Performance Oriented Usability Issues. Revista Romana de Interactiune Om-Calculator. 9(2):155-176.

Guru99 (2020). How to Use JMeter for Performance & Load Testing. Retrieved from https://www.guru99.com/jmeter-performance-testing.html on June 15, 2020.

Hamed O. & Kafri N. (2009). Performance Prediction of Web Based Application Architectures Case Study:.NET vs. Java EE. International Journal of Web Applications. 1(3): 146-156.

Jha N. & Popli R. (2017). Comparative analysis of web applications using JMeter. International Journal of Advanced Research in Computer Science. 8(3).

Oracle (2020). Command Line Reference. Retrieved from https://docs.oracle.com/cd/E13150_01/jrockit_jvm/jrockit/jrdocs/refman/optionX.html on June 15, 2020.

Oracle (2020). Java Platform, Standard Edition HotSpot Virtual Machine Garbage Collection Tuning Guide. Retrieved from https://docs.oracle.com/ javase/8/docs/technotes/guides/vm/gctuning/considerations.html on June 15, 2020.

Oracle (2020). VisualVM home page. Retrieved from https://visualvm.github.io on June 15, 2020.

Perforce Software, Inc (2020). Best Java frameworks. Retrieved from https://www.jrebel.com/blog/best-java-frameworks on June 15, 2020.

Potix (2020). Faster And Lighter, The Performance Test On ZK8. Retrieved from https://www.zkoss.org/wiki/Small_Talks/2015/September/Faster_And_Lighter,_The_Performance_Test_On_ZK8 on June 15, 2020.

Potix (2020). Our Story. Retrieved from https://www.zkoss.org/support/about on June 15, 2020.

Potix (2020). ZK home page. Retrieved from https://www.zkoss.org, on June 15, 2020.

PrimeTek Informatics (2020). Choose a Product to View Available Templates. Retrieved from https://www.primefaces.org/store on June 15, 2020.

PrimeTek Informatics (2020). Prime faces home page. Retrieved from https://www.primefaces.org on June 15, 2020.

Qing S. (2012). Web Performance Testing with Apache Jmeter. Intelligent Computer and Applications, 2.

RedHat (2020). RichFaces home page. Retrieved from https://richfaces.jboss.org on June 15, 2020.

Scholtz B. & Tijms A. (2018). The Definitive Guide to JSF in Java EE 8: Building Web Applications with JavaServer Faces. Apress.

Shan T.C. & Hua W.W. (2006). Taxonomy of java web application frameworks. In 2006 IEEE International Conference on e-Business Engineering (ICEBE'06). IEEE. pp. 378-385.

SoftwareTestingHelp (2020). Performance Testing Vs Load Testing Vs Stress Testing (Difference). Retrieved from https://www.softwaretestinghelp.com/what-is-performance-testing-load-testing-stress-testing on June 15, 2020.

The Apache Software Foundation (2020). Apache Jmeter home page. Retrieved from https://jmeter.apache.org on June 15, 2020.

The Apache Software Foundation (2020). Apache Tomcat 9 Configuration Reference. Retrieved from https://tomcat.apache.org/tomcat-9.0-doc/config/http.html on June 15, 2020.

Try QA (2020). What is Load testing in software testing? Examples, How To Do, Importance, Differences. Retrieved from http://tryqa.com/what-is-load-testing-in-software on June 15, 2020.

Vosloo I., & Kourie D.G. (2008). Server-centric web frameworks: An overview. ACM Computing Surveys (CSUR). 40(2): 1-33.