

## LỜI CẢM ƠN

Lời đầu tiên tôi xin chân thành gửi lời cảm ơn đến thầy Ngô Bá Hùng. Trong thời gian qua, thầy đã dành nhiều thời gian và tâm huyết hướng dẫn tận tình cho tôi cũng những sự hỗ trợ về nhiều mảng kiến thức, những định hướng, những giải đáp vấn đề khó khăn trong quá trình làm luận văn để tôi có thể hoàn thành đề tài luận văn tốt nghiệp một cách tốt nhất.

Tôi cũng xin gửi lời cảm ơn đến quý thầy, cô tổ kỹ thuật Khoa Công nghệ Thông tin, trong suốt thời gian làm luận văn đã luôn kề bên hỗ trợ cho tôi về kiến thức, trang thiết bị để tôi có thể thuận lợi tìm hiểu và nghiên cứu trong tin thần thoải mái nhất.

Bên cạnh đó, tôi càng chân thành gửi lời cảm ơn tha thiết đến quý thầy, cô Khoa Công nghệ Thông tin và Truyền thông, Trường Đại học Cần Thơ. Thầy, cô đã tận tình truyền dạy những kiến thức quý báu trong học tập, những kiến thức đó là những nền tảng và những kinh nghiệm thực tế trong cuộc sống trang bị cho tôi hành trang quý giá vào đời.

Tôi cũng xin gửi lời cảm ơn đến những người bạn của tôi. Các bạn đã giúp đỡ tôi rất nhiều trong việc tìm kiếm tài liệu, lời khuyên hữu ích, chia sẻ những niềm vui và rắc rối trong quá trình hoàn thành luận văn. Cuối cùng, tôi xin cảm ơn đến gia đình và người thân đã luôn tiếp thêm nguồn động và tinh thần cho tôi, giúp đỡ, chăm sóc tôi trong suốt quá trình học tập và thời gian thực hiện luận văn này.

Xin gửi lời cảm ơn chân thành đến tất cả.

Sinh viên  
**Đinh Hữu Nhân**

**NHẬN XÉT GIẢNG VIÊN HƯỚNG DẪN**

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Cần Thơ, ngày ....tháng....năm 2015

**Giảng viên hướng dẫn**

**TS. Ngô Bá Hùng**

**NHẬN XÉT CỦA HỘI ĐỒNG PHẢN BIỆN**

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Cần thơ, ngày ....tháng....năm 2015

**Hội đồng phản biện**

## MỤC LỤC

LỜI CẢM ƠN .....	i
MỤC LỤC.....	iv
DANH MỤC HÌNH ẢNH, BIỂU BẢNG, BIỂU ĐỒ .....	vii
TÓM TẮT .....	ix
ABSTRACT.....	x
KÝ HIỆU VÀ VIẾT TẮT.....	xi
PHẦN GIỚI THIỆU .....	1
I. ĐẶT VẤN ĐỀ .....	1
II. NHỮNG NGHIÊN CỨU LIÊN QUAN.....	1
III. MỤC TIÊU ĐỀ TÀI.....	2
IV. ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU.....	2
4.1. Đối tượng nghiên cứu .....	2
4.2. Phạm vi nghiên cứu.....	2
V. PHƯƠNG PHÁP NGHIÊN CỨU.....	2
VI. NỘI DUNG NGHIÊN CỨU .....	3
VII. BỐ CỤC LUẬN VĂN.....	4
PHẦN NỘI DUNG .....	5
CHƯƠNG 1: MÔ TẢ BÀI TOÁN .....	5
1.1. MÔ TẢ CHI TIẾT BÀI TOÁN .....	5
1.2. PHÂN TÍCH CÁC GIẢI PHÁP.....	6
1.2.1. Các giải pháp chia tải Cluster server.....	7
1.2.2. Quản lý session trong cluster server .....	7
1.2.3. Giải pháp sử dụng CSDL NoSQL.....	7
CHƯƠNG 2: THIẾT KẾ VÀ CÀI ĐẶT GIẢI PHÁP.....	9
2.1. THIẾT KẾ GIẢI PHÁP .....	9
2.1.1. KIẾN TRÚC TỔNG THỂ CỦA HỆ THỐNG.....	10
2.1.2. TỔNG QUAN VỀ CLUSTER.....	11
2.1.3. NGINX SERVER .....	13
2.1.4. LOAD BALANCING VỚI NGINX.....	17

2.1.5. TOMCAT SERVER .....	18
2.1.6. NoSQL .....	20
2.1.7. MONGODB .....	22
2.1.8. REPLICATION MONGODB .....	24
2.1.9. SHARDING MONGODB .....	27
2.1.10. CÔNG CỤ APACHE JMETER.....	28
2.2. CÀI ĐẶT GIẢI PHÁP.....	30
2.2.1. MÔI TRƯỜNG CÀI ĐẶT CLUSTER.....	30
2.2.2. CẤU HÌNH REVERSE PROXY .....	31
2.2.3. LOAD BALANCING VỚI NGINX.....	32
2.2.4. CẤU HÌNH REPLICATION SESSION VỚI TOMCAT SERVER .....	34
2.2.5. SHARDING MONGODB .....	35
CHƯƠNG 3: KIỂM THỬ VÀ ĐÁNH GIÁ .....	37
3.1. MỤC TIÊU KIỂM THỬ.....	37
3.2. KỊCH BẢN KIỂM THỬ.....	37
3.2.1. Kịch bản mô phỏng đăng ký học phần.....	37
3.2.2. <b>Cài đặt Apache Jmeter theo kịch bản</b> .....	41
3.3. KẾT QUẢ KIỂM THỬ.....	47
3.3.1. Quá trình thực hiện.....	47
3.3.2. Nhận xét chung .....	53
PHẦN: KẾT LUẬN.....	54
I. KẾT QUẢ ĐẠT ĐƯỢC .....	54
1.1. Kết quả .....	54
1.2. Hạn chế.....	54
II. HƯỚNG PHÁT TRIỂN .....	54
TÀI LIỆU THAM KHẢO.....	56
PHỤ LỤC .....	57
PHỤ LỤC A: MONGODB .....	57
1. Cài đặt MongoDB trên Ubuntu Server 14.04 .....	57
2. Index.....	57

3. Truy vấn .....	59
4. Hướng dẫn replication mongodb .....	63
5. Cấu hình sharding với mongodb .....	65
PHỤ LỤC B: NGINX SERVER.....	66
1. Cài đặt Nginx .....	66
2. Cấu hình cơ bản Nginx .....	66
3. Các thông số của Nginx .....	66
4. Cấu hình liên quan tới Http .....	69
5. Cấu hình Reverse Proxy và Load balancing .....	71
6. Cấu hình replication session Tomcat Server 7.....	72
PHỤ LỤC C: CÀI ĐẶT VÀ THIẾT LẬP MÔI TRƯỜNG JAVA.....	75
1. Cài đặt JDK trên Ubuntu server 14.04.....	75
2. Cài đặt Tomcat server 7 .....	75
3. Cài đặt Apache Jmeter .....	75

## **DANH MỤC HÌNH ẢNH, BIỂU BẢNG, BIỂU ĐỒ**

### **DANH MỤC HÌNH ẢNH**

Hình 1: Mô hình ứng dụng phân tán đa tầng .....	9
Hình 2: Kiến trúc tổng thể của hệ thống .....	10
Hình 3: Cơ chế hoạt động của cluster .....	12
Hình 4: Kiến trúc tiến trình Nginx .....	15
Hình 5: Cơ chế hoạt động Reverse Proxy .....	16
Hình 6: Replication Session .....	19
Hình 7: Cơ chế hoạt động replication session .....	19
Hình 8: Cấu trúc một document .....	23
Hình 9: So sánh RDBMS với MongoDB .....	24
Hình 10: Mô hình Master – Slave hai nút .....	25
Hình 11: Mô hình Master – Slave bốn nút .....	25
Hình 12: Mô hình Replica Sets hai nút .....	26
Hình 13: Replica Sets – Bầu chọn master mới .....	26
Hình 14: Server chính trở thành server cấp 2 .....	27
Hình 15: Cơ chế Sharding .....	27
Hình 16: Balancing MongoDB .....	28
Hình 17: Mô hình triển khai reverse proxy .....	31
Hình 18: Mô hình triển khai load balancing .....	33
Hình 19: Mô hình cài đặt Session Replication .....	34
Hình 20: Mô hình triển khai MongoDB .....	36
Hình 21: Test Plan .....	42
Hình 22: Thread group .....	42
Hình 23: HTTP Cookie Manager .....	43
Hình 24: HTTP Request Defaults .....	43
Hình 25: HTTP Request Home .....	44
Hình 26: CSV Data set Config .....	44
Hình 27: HTTP Request login .....	45
Hình 28: HTTP Request xem đăng ký mã học phần CT333 .....	45
Hình 29: HTTP Request đăng ký mã học phần CT333 .....	46
Hình 30: HTTP Request xóa mã học phần CT333 .....	46

### **DANH MỤC BIỂU BẢNG**

Bảng 1: Bảng phân công công việc .....	4
Bảng 2: So sánh CSDL quan hệ và NoSQL .....	8
Bảng 3: So sánh Nginx với Apache .....	14
Bảng 4: Cơ sở hạ tầng phần mềm .....	31

Bảng 5: Cấu hình cluster MongoDB .....	36
Bảng 6: Kịch bản kiểm thử .....	40
Bảng 7: Thông tin server sử dụng một Tomcat server.....	47
Bảng 8: Kết quả thu được sử dụng một Tomcat server .....	47
Bảng 9: Bảng thông tin server khi sử dụng 2 tomcat server .....	49
Bảng 10: Kết quả trả về của kịch bản 400 user cùng lúc sử dụng hai tomcat server.....	50
Bảng 11: Thông tin server sử dụng 2 tomcat server với 800 user .....	51
Bảng 12: Kết quả trả về của kịch bản 800 user cùng lúc sử dụng hai tomcat server.....	52

## **DANH MỤC BIỂU ĐỒ**

Biểu đồ 1: kết quả 400 user cùng lúc sử dụng một tomcat server .....	48
Biểu đồ 2: Thời gian xử lý request 400 user cùng lúc sử dụng một Tomcat server .....	48
Biểu đồ 3: Kết quả 400 user cùng lúc sử dụng hai tomcat server.....	50
Biểu đồ 4: Thời gian xử lý request 400 user cùng lúc sử dụng hai Tomcat server .....	50
Biểu đồ 5: Kết quả 800 user cùng lúc sử dụng hai tomcat server.....	52
Biểu đồ 6: Thời gian xử lý request 800 user cùng lúc sử dụng hai Tomcat server .....	52



## TÓM TẮT



Sự phát triển nhanh chóng của công nghệ mạng Internet và công nghệ web, cùng với sự áp dụng hệ thống học theo tín chỉ của các trường đại học, xây dựng website đăng ký học phần trực tuyến là điều tất yếu nhằm đơn giản hóa trong công việc quản lý sinh viên đăng ký học phần.

Hiện nay, số lượng sinh viên của các trường đại học khá lớn (lên đến vài chục ngàn người), do đó có lượng lớn truy cập server cùng lúc gặp phải một số khó khăn sau:

- Hệ thống web server không thể nào đáp ứng được hết tất cả yêu cầu của các sinh viên đăng ký học phần cùng một lúc.
- Mỗi khi đăng ký học phần, server CSDL phải làm việc với công suất tối đa (dẫn đến tình trạng server CSDL quá tải) vì việc đọc, ghi dữ liệu quá nhiều.

Từ những khó khăn trên, để đáp ứng được một lượng sinh viên đăng ký học phần cùng một lúc, nên chúng tôi muốn thực hiện một số thử nghiệm sau:

- Từ việc sử dụng một web server không thể nào đáp ứng được tất cả các yêu cầu trong thời gian nhanh nhất. Nên chúng tôi áp dụng thử nghiệm xây dựng hệ thống cluster có khả năng chịu tải cao và khả năng mở rộng chính là việc cân bằng tải cho hệ thống web sever.
- Mỗi năm, các trường đại học phải lưu thêm một lượng lớn dữ liệu vào máy chủ CSDL quan hệ, đến một thời gian nào đó thì thời gian truy xuất dữ liệu của máy chủ trở nên chậm chạp. Chính vì thế chúng tôi muốn thử nghiệm sử dụng CSDL NoSQL thay vì sử dụng CSDL quan hệ. Do CSDL NoSQL có thể lưu trữ được một lượng lớn dữ liệu, và tốc độ truy xuất dữ liệu khá nhanh.

## ABSTRACT



The rapid development of Internet technology and web technology, along with the application of the system of credits universities, Therefore, building website online registration module is indispensable to simplify the registered management module.

Currently, the number of college students is quite large (up to more than tens thousands of people), so there is huge access at the same time lead to some difficulties following:

- Web server system can not meet all requirements of the students simultaneously.
- When having registration, the database server has to work with a maximum capacity (leading to a database server overload) for reading, writing too much data.

From these difficulties, to meet a large number of students enroll at the same time, so we try to perform some tests:

- When using a web server can not meet all the requirements in the shortest possible time. Therefore, we apply building cluster system test with high load-bearing capacity and balancing load for web sever systems.
- Every year, the universities have to save large quantities of data into a relational database server, to a certain time, the data access time of the server becomes slow. Therefore we want to test using NoSQL database instead of using a relational database. Because NoSQL database can store large amounts of data, and the speed of data access is fast confidently.

### KÝ HIỆU VÀ VIẾT TẮT

Từ/kí hiệu viết tắt	Nguyên bản
CSDL	Cơ sở dữ liệu
NoSQL	Not Only SQL
RDBMS	Relational database management
JSON	JavaScript Object Noation

## PHẦN GIỚI THIỆU

### I. ĐẶT VẤN ĐỀ

Hiện nay, các trường đại học đã áp dụng phương pháp đào tạo theo tín chỉ, hầu hết một số trường đều đã triển khai hệ thống đăng ký học phần trực tuyến. Số lượng sinh viên của mỗi trường đại học, cao đẳng là khá lớn, nên số lượng truy cập vào website tăng lên. Do đó, server không thể phục vụ được hết một lượng lớn các yêu cầu, để giải quyết nhanh vấn đề trên, trường phải chia ra thành nhiều nhóm khác nhau để đăng ký nhằm giảm tải cho server.

Trong quá trình đăng ký với số lượng lớn truy cập cùng lúc vào website có thể xảy ra một số vấn đề sau:

- Web server không thể đủ khả năng xử lý tất cả các yêu cầu cùng lúc.
- Server CSDL không thể đáp ứng nhanh các truy vấn từ web server.

Từ những vấn đề trên, chúng tôi xây dựng một hệ thống có khả năng chịu tải cao và chọn CSDL NoSQL thay thế cho CSDL SQL (quan hệ).

Xuất phát từ nhu cầu thực tế, chúng tôi nghiên cứu hệ thống có khả năng đáp ứng được số lượng sinh viên đăng ký học phần cùng lúc. Chính vì thế, tôi và bạn Lê Huy Hoàng quyết định chọn đề tài “Phát triển đăng ký môn học hướng thông lượng người dùng lớn” do thầy TS. Ngô Bá Hùng hướng dẫn.

### II. NHỮNG NGHIÊN CỨU LIÊN QUAN

Đề tài luận văn khoa Công nghệ Thông tin và Truyền thông Đại học Cần Thơ: Đánh giá tải Moodle của Quách Kim Hải.

Hệ thống đăng ký học phần hiện tại đã đáp ứng đầy đủ hầu như tất cả các yêu cầu trong công tác quản lý sinh viên đăng ký học phần. Một số chức năng của website đăng ký học phần nay cũng đã được hoàn chỉnh, ví dụ như: Tìm kiếm lớp học phần, đăng ký học phần, xem thời khóa biểu,...

Song đó, hệ thống còn một số khó khăn cho nhà quản lý và sinh viên là: phải chia ra nhiều đợt đăng ký học phần khác (ví dụ: thời gian đăng ký của từng khoa sẽ khác nhau) để giảm tải cho server.

### III. MỤC TIÊU ĐỀ TÀI

Mục tiêu chung của nhóm đề tài là nghiên cứu xây dựng website đăng ký học phần hướng thông lượng người dùng lớn.

Các mục tiêu cụ thể như sau:

1. Tìm hiểu về Spring MVC xây dựng website đăng ký học phần sử dụng cơ sở dữ liệu MongoDB.
2. Tìm hiểu về cơ sở dữ liệu với MongoDB và cách thiết kế CSDL với MongoDB.
3. Xây dựng hệ thống cluster cho phép chịu tải cao dùng máy chủ Nginx và cơ chế load balancing, reverse proxy với Nginx, quản lý session với Tomcat.
4. Tìm hiểu các tính năng nâng cao của MongoDB: Index, Replication, Shard,...
5. Tìm hiểu phần mềm Apache Jmeter để giả lập người dùng đăng ký học phần.

Trọng tâm của đề tài này là mục tiêu 1,2,3,4.

### IV. ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU

#### 4.1. Đối tượng nghiên cứu

- Hệ thống đăng ký học phần trực tuyến của trường Đại học Cần Thơ.
- Thiết kế CSDL với MongoDB.
- Tính năng Replication, Shard của MongoDB.
- Xây dựng hệ thống cluster: Nginx, Tomcat, MongoDB.
- Công cụ hỗ trợ kiểm thử ứng dụng web Apache JMeter.

#### 4.2. Phạm vi nghiên cứu

- Xây dựng website đăng ký học phần dựa Spring MVC – MongoDB. Ở đề tài này, chúng tôi chỉ chú trọng tâm phần đăng ký học phần (các dữ liệu như kế hoạch học tập, danh sách lớp học phần, .... Được xem như là có sẵn và được import vào hệ thống)
- Xây dựng hệ thống cluster với Nginx, Tomcat, MongoDB.

### V. PHƯƠNG PHÁP NGHIÊN CỨU

- Nghiên cứu hệ thống đăng ký học phần của trường đại học Cần Thơ.
- Nghiên cứu trọng tâm về cluster và NoSQL.

- Tự tìm hiểu đề tài thông qua nhiều nguồn khác nhau: internet, các nghiên cứu, đề tài liên quan, tài liệu từ giảng viên hướng dẫn. Lập kế hoạch thực hiện.
- Làm việc nhóm một cách hiệu quả: phân chia thời gian, công việc hợp lý, lập kế hoạch làm việc cho mỗi thành viên, chia sẻ kiến thức, giúp đỡ lẫn nhau để hoàn thành công việc.
- Nhờ vào sự giúp đỡ, gợi ý của giảng viên hướng dẫn khi có vấn đề khó giải quyết hoặc chưa nắm rõ cần được giải thích thêm.

## VI. NỘI DUNG NGHIÊN CỨU

Đề tài được chia thành 2 phần hệ:

- Phân hệ 1: Xây dựng website đăng ký học phần dựa trên công nghệ Spring MVC với CSDL MongoDB. Thiết kế CSDL với MongoDB.
- Phân hệ 2: Xây dựng hệ thống cluster, cơ chế cân bằng tải với Nginx Server và thực hiện cơ chế phân tán dữ liệu (shard) với MongoDB.

Bảng phân công công việc:

Thời gian (Tuần)	Đinh Hữu Nhân	Lê Huy Hoàng
1,2	Tìm hiểu yêu cầu đề tài và các vấn đề liên quan khác.	
3,4,5	Tìm hiểu về MongoDB và thiết kế CSDL cho website.	
6,7	Tìm hiểu và cài đặt hệ thống Nginx Server.	Tìm hiểu công nghệ Spring Framework và Maven.
8,9	Cấu hình load balancing và reverse proxy cho Tomcat với Nginx.	Tìm hiểu Spring MVC với MongoDB.
10,12	Cấu hình Replication Session Tomcat Server.	Thiết kế giao diện website với Bootstrap.
13,14	Cấu hình Replication và Shard với MongoDB.	Sử dụng Spring MVC để xử lý các chức năng của website.
14,15	Đánh giá và kiểm thử: sử dụng tool Apache JMeter giả lập người dùng và đánh giá hệ thống thông qua Nagios Server.	
16	Viết báo cáo.	

17	Báo cáo.
----	----------

*Bảng 1: Bảng phân công công việc*

## VII. BỐ CỤC LUẬN VĂN

Luận văn được chia thành 3 chương:

- **Chương 1: Mô tả bài toán**

Ở chương này, chúng tôi sẽ mô tả chi tiết bài toán, phân tích các giải pháp khi xây dựng cluster, lựa chọn giải pháp cân bằng tải, quản lý session trong hệ thống cluster, thay thế CSDL quan hệ bằng NoSQL.

- **Chương 2: Thiết kế và cài đặt giải pháp**

- **Thiết kế giải pháp**

Chương này đề xuất mô hình hệ thống cluster, giới thiệu và giải thích cơ chế hoạt động của từng thành phần (Nginx, Tomcat, MongoDB) trong hệ thống. Giới thiệu công cụ kiểm thử hiệu suất Apache JMeter.

- **Cài đặt giải pháp**

Dựa trên những giải pháp đã thiết kế, chương này trình bày môi trường cài đặt cho hệ thống cluster. Cài đặt và cấu hình cho hệ thống cluster theo giải pháp đã thiết kế.

- **Chương 3: Kiểm thử và đánh giá**

Chương này chúng ta đưa ra được các mục tiêu kiểm thử, kịch bản kiểm thử, sử dụng công cụ JMeter để giả lập người dùng. Nhận kết quả và đánh giá hệ thống cluster.

## PHẦN NỘI DUNG

### CHƯƠNG 1: MÔ TẢ BÀI TOÁN

#### 1.1. MÔ TẢ CHI TIẾT BÀI TOÁN

Một số trường đại học áp dụng chế độ học theo tín chỉ và cho phép sinh viên có quyền lựa chọn môn học cho mỗi học kỳ. Dựa vào kế hoạch đào tạo và dựa vào chương trình khung của từng ngành, hệ thống lập thời khóa biểu dự kiến cho từng môn học của từng ngành trong một học kỳ.

Trước khi bước vào học kỳ mới các giảng viên đăng ký các môn mà mình có thể dạy trong học kỳ đó. Căn cứ vào kế hoạch đào tạo và thời khóa biểu dự kiến đã lập, hệ thống hỗ trợ việc hiển thị lịch học dự kiến cho từng ngành trong từng học kỳ, danh sách các học phần bắt buộc và tự chọn dự kiến sẽ dạy, đề cương chi tiết, điều kiện tiên quyết, số tín chỉ, thời gian học, thời lượng học, số lượng sinh viên tối đa được phép, số lượng sinh viên hiện tại đã đăng kí để sinh viên có căn cứ lựa chọn.

Sinh viên đăng ký tối đa 20 tín chỉ cho mỗi học kỳ và việc đăng ký được thực hiện trong thời gian mở đăng ký học phần. Các môn được cung cấp cho sinh viên là các môn mà nhà trường dự kiến đào tạo nằm trong khung chương trình của Ngành. Việc đăng ký các môn học cho từng học kỳ phải bảo đảm điều kiện tiên quyết của từng học phần và trình tự học tập của mỗi chương trình cụ thể. Sau đây là một số chức năng cơ bản của website đăng ký học phần:

- Đăng nhập vào hệ thống.
- Xem thông tin sinh viên.
- Xem danh mục học phần.
- Đăng ký học phần (đăng ký, sửa, xóa).
- Xem thời khóa biểu.

Bên cạnh những thuận tiện cho sinh viên đăng ký học phần và đơn giản cho công tác quản lý của cán bộ. Hệ thống còn một số khó khăn cần cải tiến:

- Thời khóa biểu của lớp học phần phải được ra thành nhiều đợt khác nhau, mỗi đợt học phải có nhiều buổi học.
- Tăng số lượng người dùng truy cập đồng thời vào website đăng ký học phần.

Ngày nay, với sự bùng nổ số lượng người dùng sử dụng Internet đặc biệt là ứng dụng web, thương mại điện tử... cùng với sự phát triển của các trang web có nội dung động (dynamic content) đã làm gia tăng đáng kể khả năng xử lý của server. Đến thời



điểm nào đó sẽ không thể đáp ứng với số lượng lớn các yêu cầu đồng thời, và giải pháp đưa ra là thay thế hoặc nâng cấp server khác cấu hình mạnh, khả năng xử lý cao nhưng giải pháp này không khả thi vì lý do sau đây:

- Với server mới được thay thế này thì trong tương lai gần lại không thể đáp ứng được số lượng các yêu cầu lớn hơn và chúng ta lại thay thế hoặc nâng cấp server này bằng các yêu cầu lớn hơn và chúng ta lại phải thay thế hoặc nâng cấp server này bằng server khác có cấu hình mạnh mẽ hơn và điều này sẽ còn tiếp tục đến khi nào?
- Với mỗi công đoạn thay thế hay nâng cấp server thì chi phí trả cho thiết bị phần cứng khá cao, và giải pháp này xem ra không mang lại hiệu quả kinh tế cho lắm.

Từ các vấn đề trên, chúng ta đưa ra khái niệm: “Cluster server”. Để đảm bảo tính sẵn sàng của server cao, có thể nâng cấp hệ thống cluster dễ dàng.

Trong hệ thống cluster bao gồm:

- Nginx server: có chức năng load balancer, reverse proxy.
- Tomcat server: là application server.
- MongoDB: chức năng lưu trữ dữ liệu.

## 1.2. PHÂN TÍCH CÁC GIẢI PHÁP

Từ bài toán trên, để giải quyết vấn đề gia tăng đáng kể các yêu cầu từ client các nhà quản trị có thể dùng 2 giải pháp sau đây:

- *Single Server*: Thay thế server cũ (hardware) bằng server mới, mạnh hơn và nhanh hơn có thể đáp ứng được tốt các yêu cầu. Mặc dù server hiện tại có thể mạnh nhưng không thể đáp ứng được một lượng lớn yêu cầu web động. Hơn nữa, giải pháp single server mang lại khuyết điểm: giá thành cao (đầu tư server mạnh), không có khả năng mở rộng hệ thống, không có tính chịu lỗi cao, ...
- *Cluster Server*: là một nhóm server hoạt động cùng nhau để chia tải công việc, cung cấp độ tin cậy và khả năng xử lý cao cho hệ thống. Đối với client, cluster server hoạt động giống như một server đơn lẻ nhưng thật chất nó là nhóm nhiều server.

Trong đề tài, chúng tôi bàn về giải pháp là cluster server, giải pháp này cho phép hệ thống cluster server của ta đáp ứng được một lượng rất lớn các yêu cầu đồng thời hơn single server.

### 1.2.1. Các giải pháp chia tải Cluster server

Trên mô hình client/server, một phía là client phía còn lại là server và có thể tồn tại proxy ở giữa như proxy server cho các dịch vụ web. Dựa vào đặc điểm này, chúng ta có nhiều cách điều phối yêu cầu khác nhau. Thông thường các server cluster sẽ chạy cùng dịch vụ và cùng một nội dung cung cấp. Các giải pháp đề nghị cho việc phân tán các yêu cầu phục vụ có thể kể ra như sau:

- *Hướng xây dựng phía client*: cung cấp một applet chạy phía client, khi đó applet tạo ra các yêu cầu đến cluster và lấy thông tin trạng thái của cluster, sau đó lựa chọn server phục vụ cho yêu cầu của mình dựa vào thông tin chạy thái đó và forward yêu cầu đó đến server.
- *Hướng xây dựng phía server* (Round Robin - Nginx): Mỗi lượt truy cập khác nhau của client sẽ được chuyển đến các server khác nhau trong cluster, phân tán yêu cầu cho các server.

Ở giải pháp chia tải cho cluster server chúng tôi chọn giải pháp “Hướng xây dựng phía server (“Round Robin - Nginx”)”. Giải pháp có thể làm giảm đi độ phức tạp của hệ thống, không cần phía client phải chạy applet nào cả và việc cấu hình load balancing với Nginx tương đối đơn giản.

### 1.2.2. Quản lý session trong cluster server

Việc sử dụng Nginx làm load balancer cho các server Tomcat, sẽ xảy ra vấn đề là khi yêu cầu của ta được chuyển đến một Tomcat Server khác trong hệ thống cluster, nhưng session không được replicated trước đó thì session của ta sẽ bị mất và dẫn đến lỗi. Đến đây ta thấy công nghệ session replication thật sự cần thiết cho môi trường clustering.

### 1.2.3. Giải pháp sử dụng CSDL NoSQL

Mỗi năm, server CSDL của website đăng ký học phần được lưu trữ thêm vào một lượng lớn dữ liệu (thông tin sinh viên, thời khóa biểu, lớp học phần, ...) và tăng lên không ngừng. Đến một thời gian nào đó, ứng dụng web của chúng ta chứa một lượng dữ liệu khổng lồ sẽ có rất nhiều khó khăn để đạt khả năng xử lý như mong muốn đối với hệ quản trị dữ liệu ràng buộc (RDBMS) nhưng lại được giải quyết bằng NoSQL.

**Tại sao chọn CSDL NoSQL:**

Tính năng	CSDL quan hệ	NoSQL
-----------	--------------	-------

Hiệu suất	Kém hơn SQL Relational giữa các table	Cực tốt Bỏ qua SQL Bỏ qua các ràng buộc dữ liệu
Khả năng mở rộng	Hạn chế về lượng.	Hỗ trợ một lượng rất lớn các
Hiệu suất đọc-ghi	Kém do thiết kế để đảm bảo sự vào/ra liên tục của dữ liệu	Tốt với mô hình xử lý lô và những tối ưu về đọc-ghi dữ liệu.
Thay đổi số node trong hệ thống	Phải shutdown cả hệ thống. Việc thay đổi số node phức tạp.	Không cần phải shutdown cả hệ thống. Việc thay đổi số node đơn giản, không ảnh hưởng đến hệ thống.
Phần cứng	Đòi hỏi cao về phần cứng.	Đòi hỏi thấp hơn về giá trị và tính đồng nhất của phần cứng

*Bảng 2: So sánh CSDL quan hệ và NoSQL*

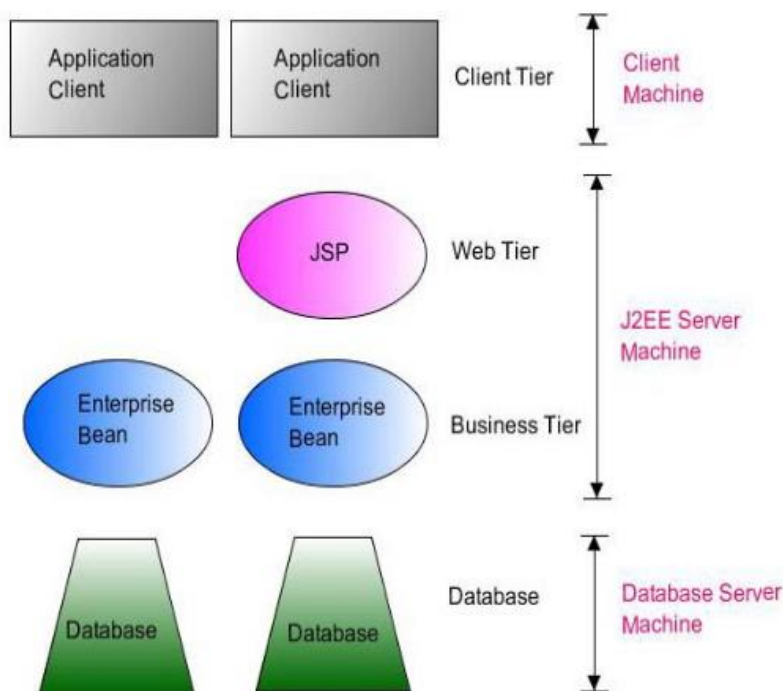
## CHƯƠNG 2: THIẾT KẾ VÀ CÀI ĐẶT GIẢI PHÁP

Chương này sẽ trình bày cách thiết kế và xây dựng mô hình hệ thống cluster, giải thích rõ các thành phần trong cluster. Nêu rõ đặc điểm, cách thức hoạt động của từng thành phần trong hệ thống cluster tạo tiền đề để cài đặt hệ thống.

Để tăng tính chịu tải cao của hệ thống cluster, cần cài đặt một hệ thống cluster gồm Nginx, Tomcat Server, sử dụng cơ sở dữ liệu là MongoDB. Phần này hướng đến phần cách cài đặt một hệ thống cluster.

### 2.1. THIẾT KẾ GIẢI PHÁP

J2EE nền tảng sử dụng một mô hình ứng dụng phân tán đa tầng.



Hình 1: Mô hình ứng dụng phân tán đa tầng

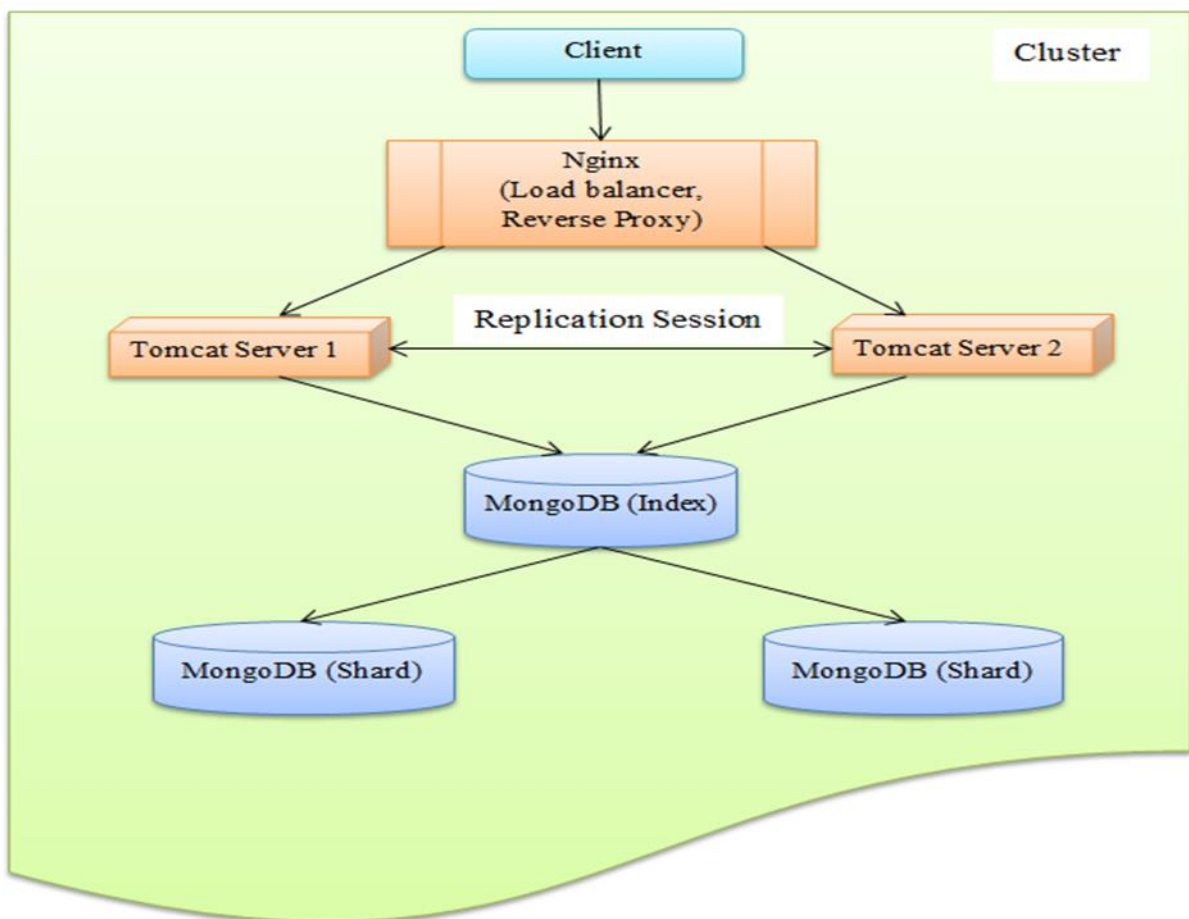
Trong mô hình ứng dụng J2EE có nhiều tầng: Tầng khách hàng (client tier), tầng web (web tier), tầng thương mại (business tier) và tầng hệ thống thông tin thương mại (enterprise information system tier). Tầng thương mại và tầng web nằm trên một máy chủ ứng dụng gọi là máy chủ ứng dụng (application server) hay máy chủ J2EE (J2EE server). Máy chủ J2EE cung cấp những dịch vụ cần thiết cho những thành phần

(component) của tầng thương mại và tầng web. Từ mô hình trên chúng ta có thể triển khai kiến trúc tổng thể của hệ thống.

### 2.1.1. KIẾN TRÚC TỔNG THỂ CỦA HỆ THỐNG

Mô hình cluster gồm:

- **Nginx Server:** có chức năng làm Reverse Proxy và Load Balancer.
- **Tomcat server:** xử lý các request từ client.
- **MongoDB (Index):** có chức năng như là một Load Balancer (Routing) cho các Shard phía sau.
- **MongoDB (Shard):** có chức năng thêm, cập nhật, xóa cơ sở dữ liệu.



Hình 2: Kiến trúc tổng thể của hệ thống

## 2.1.2. TỔNG QUAN VỀ CLUSTER

### 2.1.2.1. Khái niệm cluster

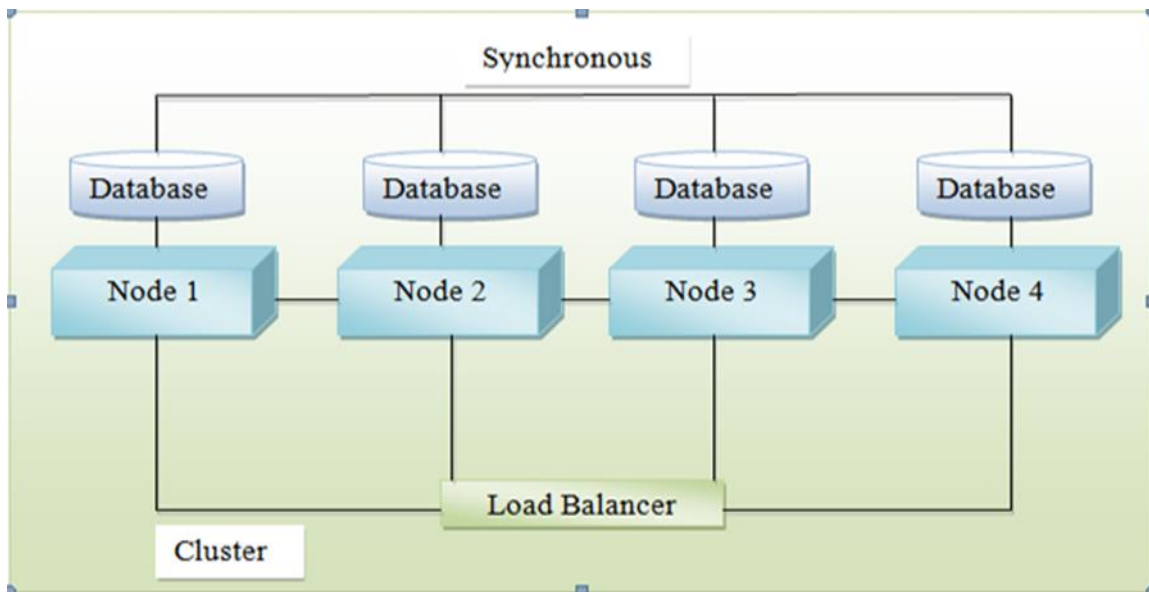
Cluster là một kiến trúc nhằm đảm bảo nâng cao khả năng sẵn sàng cho hệ thống máy tính. Cluster cho phép nhiều máy chủ kết hợp với nhau tạo thành một cụm có khả năng chịu đựng hay chấp nhận sai sót nhằm nâng cao độ sẵn sàng của hệ thống máy tính. Cluster là một hệ thống bao gồm nhiều máy chủ được kết hợp với nhau theo dạng song song hay phân tán và được sử dụng một tài nguyên thống nhất. Nếu một máy chủ ngừng hoạt động do bị sự cố hoặc để nâng cấp, bảo trì thì toàn bộ công việc mà máy chủ này đảm nhận sẽ được tự động chuyển cho một máy chủ khác (trong cùng một cluster) mà không làm cho hoạt động của hệ thống bị ngắt, gián đoạn hoặc trì trệ hệ thống.

Việc thiết kế và cài đặt các cluster cần thoả mãn các yêu cầu sau:

- **Yêu cầu tính sẵn sàng cao:** Các tài nguyên mạng phải luôn sẵn sàng trong khả năng cao nhất để cung cấp và phục vụ người dùng cuối và giảm thiểu sự ngưng hoạt động hệ thống ngoài ý muốn.
- **Yêu cầu độ tin cậy cao:** Độ tin cậy của một cluster được hiểu là khả năng giảm thiểu tần số xảy ra các sự cố, và nâng cao khả năng chịu đựng sai sót của hệ thống.
- **Yêu cầu về khả năng mở rộng được:** Hệ thống phải có khả năng dễ dàng cho việc nâng cấp, mở rộng trong tương lai. Việc nâng cấp mở rộng bao hàm cả việc thêm thiết bị, máy tính vào hệ thống để nâng cao chất lượng dịch vụ, cũng như thêm số lượng người dùng, thêm ứng dụng, dịch vụ và thêm các tài nguyên mạng khác.

### 2.1.2.2. Cơ chế hoạt động của cluster

Mỗi máy chủ trong cluster gọi là một nút (Cluster node) và có thể thiết lập ở chế độ chủ động (active) hay thụ động (passive), khi một nút (node) ở chế độ chủ động, nó sẽ chủ động xử lý các yêu cầu khi một nút (node) là thụ động, nó sẽ nằm ở chế độ phòng nóng (standby) chờ sẵn sàng thay thế cho một nút khác nếu bị hỏng.



Hình 3: Cơ chế hoạt động của cluster

Trong một cluster có nhiều nút (node) có thể kết hợp cả nút (node) chủ động và nút thụ động. Trong mô hình loại này việc quyết định một nút được cấu hình là chủ động hay thụ động rất quan trọng. Ví dụ:

Nếu một nút chủ động bị sự cố và có nút thụ động đang sẵn sàng, các ứng dụng và dịch vụ đang chạy trên nút hỏng có thể lập tức chuyển sang nút thụ động. Vì máy chủ đóng vai trò nút thụ động hiện tại chưa chạy ứng dụng hay dịch vụ gì cả nên có thể gánh toàn bộ công việc của máy chủ hỏng mà không ảnh hưởng gì đến các ứng dụng và dịch vụ cung cấp cho người dùng cuối (end user) ngầm định rằng các máy chủ trong cluster có cấu trúc phần cứng giống nhau.

Nếu tất cả trong máy chủ cluster là chủ động và một nút bị sự cố các ứng dụng và dịch vụ đang chạy trên máy chủ hỏng phải chuyển sang một máy chủ khác cũng đóng vai trò (node) chủ động, vì là nút (node) chủ động nên bình thường máy chủ này cũng phải đảm nhận một số ứng dụng hay một số dịch vụ gì đó khi có sự cố xảy ra thì nó sẽ phải gánh thêm công việc của máy chủ hỏng. Do vậy để đảm bảo hệ thống hoạt động bình thường kể cả khi có sự cố thì máy chủ cluster cần phải có cấu hình dự phòng để có thể gánh thêm khối lượng công việc của máy chủ khác khi cần.

Trong cấu trúc cluster mà mỗi nút (node) chủ động được dự phòng bởi một nút (node) thụ động các máy chủ cần có cấu hình sao cho với khối lượng công việc trung bình chúng sử dụng hết khoảng 50% CPU và dung lượng bộ nhớ.



Trong cấu trúc cluster mà số nút chủ động nhiều hơn số nút bị động, các máy chủ cần cấu hình tài nguyên và bộ nhớ mạnh hơn nữa có thể xử lý được khối lượng công việc cần thiết khi một nút nào đó bị hỏng.

Các nút trong cluster thường là một bộ phận của cùng một vùng domain và có thể được cấu hình là máy điều khiển vùng (domain controller) hay máy chủ thành viên. Lý tưởng nhất là mỗi cluster có nhiều nút ít nhất là 2 nút làm máy chủ điều khiển vùng và đảm nhiệm việc failover đối với những dịch vụ thiết yếu. Nếu không như vậy thì khả năng của các tài nguyên trên cluster sẽ bị phụ thuộc vào khả năng sẵn sàng của các tài nguyên trên cluster sẽ bị phụ thuộc vào khả năng sẵn sàng của máy chủ điều khiển trong vùng domain.

### **2.1.3. NGINX SERVER**

#### **2.1.3.1. Nginx là gì?**

Nginx là một máy chủ proxy ngược mã nguồn mở (open source reverse proxy server) sử dụng phổ biến giao thức HTTP, HTTPS, SMTP, POP3 và IMAP, cũng như dùng làm cân bằng tải (load balancer), HTTP cache và máy chủ web (web server). Dự án Nginx tập trung vào việc phục vụ số lượng kết nối đồng thời lớn (high concurrency), hiệu suất cao và sử dụng bộ nhớ thấp. Nginx được biết đến bởi sự ổn định cao, nhiều tính năng, cấu hình đơn giản và tiết kiệm tài nguyên.

#### **2.1.3.2. Các tính năng của Nginx**

Nginx cung cấp hàng loạt dịch vụ ẩn tượng, không phải cho chỉ các tính năng liên quan tới Http mà còn nhiều tính năng khác. Sau đây là một số tính năng:

- Tính năng cơ bản máy chủ HTTP.
- Xử lý các tập tin tĩnh và các tập tin chỉ mục, và tự động lập chỉ mục, mở các tập tin cache.
- Tăng tốc với bộ nhớ đệm đảo ngược proxy, đơn giản cân bằng tải và khả năng chịu lỗi.
- Hỗ trợ tăng tốc với bộ nhớ đệm của FastCGI, uwsgi, SCGI, các máy chủ memcached.
- Bộ lọc Modular kiến trúc bao gồm các phạm vi byte gzipping... phản ứng chunked, XSTL, SSI, và bộ lọc thay đổi kích thước hình ảnh. SSL và TLS hỗ trợ SNI.

Các tính năng của máy chủ HTTP:

- Name-based and IP-based virtual server.



- Keep-alive và pipelined hỗ trợ kết nối.
- Cấu hình linh hoạt.
- Cấu hình lại và nâng cao một thực thi mà không cần gián đoạn dịch vụ của khách hàng.
- Truy cập dạng các nhật ký, viết nhật ký đậm, đăng nhập và quay nhanh.
- Viết lại mô-dun: thay đổi URI bằng cách sử dụng biểu thức thông thường.
- Thực hiện chức năng khác nhau tùy thuộc vào địa chỉ khách hàng.
- Kiểm soát truy cập dựa trên địa chỉ IP và xác thực HTTP cơ bản.
- PUT, DELETE, MKCOL, COPY, và phương pháp MOVE.
- FLV và MP4 truyền.
- Tỷ lệ hạn chế phản ứng.
- Hạn chế số lượng kết nối đồng thời, yêu cầu đến từ cùng một địa chỉ IP.
- Hỗ trợ embedded Perl.

Tính năng mail và máy chủ Proxy:

- Người sử dụng chuyển hướng để IMAP/POP3 phụ trợ bằng cách sử dụng một máy chủ HTTP xác thực bên ngoài.
- Chứng thực người dùng bằng cách sử dụng một máy chủ HTTP xác thực bên ngoài và chuyển hướng kết nối với một phụ trợ SMTP nội bộ.

Phương pháp xác thực:

- POP3: USER/PASS, AUTH LOGIN/PLAIN/CRAM-MD5.
- IMAP: LOGIN, AUTH LOGIN/PLAIN/CRAM-MD5.
- SMTP: AUTO LOGIN/PLAIN/CRAMMD5.
- Hỗ trợ SSL.
- Hỗ trợ STARTTLS, STLS.

Nginx tương thích với nhiều kiến trúc máy tính và hệ điều hành Window, Linux, Mac OS, Free BSD.

### 2.1.3.3. So sánh Nginx và Apache

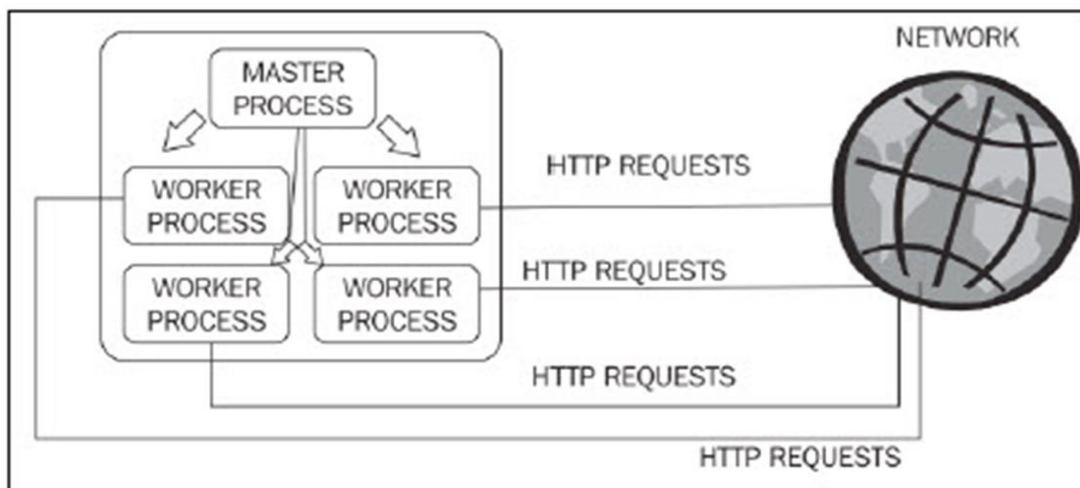
Nginx	Apache
Nginx: không dựa vào luồng (threads) để xử lý các truy vấn (request). Thay vào đó, Nginx sử dụng kiến trúc hướng sự kiện (event-driven) không đồng bộ (asynchronous) và có khả năng mở rộng.	Mỗi khi nhận yêu cầu mới, nó sẽ tạo ra một luồng để xử lý mới để xử lý yêu cầu này. Số lượng yêu cầu càng nhiều thì số lượng luồng càng tăng → làm hao tốn tài nguyên (CPU, RAM).

Bảng 3: So sánh Nginx với Apache

#### 2.1.3.4. Kiến trúc tiến trình

Mỗi khi chạy Nginx, 1 tiến trình duy nhất tồn tại trong bộ nhớ – gọi là “Tiến trình chủ” (Master Process). Tiến trình này được chạy với quyền của tài khoản và nhóm tài khoản hiện tại – thường là root/root nếu dịch vụ được chạy tại thời gian khởi động bởi script init. Tiến trình chủ tự nó không xử lý bất kỳ yêu cầu nào từ người dùng, thay vào đó, nó sinh ra các tiến trình thực hiện việc xử lý này – gọi là “Tiến trình công nhân” (Worker Process).

Từ tập tin cấu hình, chúng ta có thể định nghĩa số tiến trình công nhân, số lượng kết nối tối đa cho mỗi tiến trình công nhân, tài khoản và nhóm tài khoản mà các tiến trình công nhân chạy dưới quyền.



Hình 4: Kiến trúc tiến trình Nginx

#### 2.1.3.5. Cấu hình cơ bản Nginx

Các module cơ bản cung cấp các khai báo cho phép chúng ta định nghĩa các tham số của các chức năng cơ bản của Nginx.

Xem chi tiết tham khảo phần 2,3,4 phụ lục B.

### 2.1.4. PROXY REVERSE VỚI NGINX

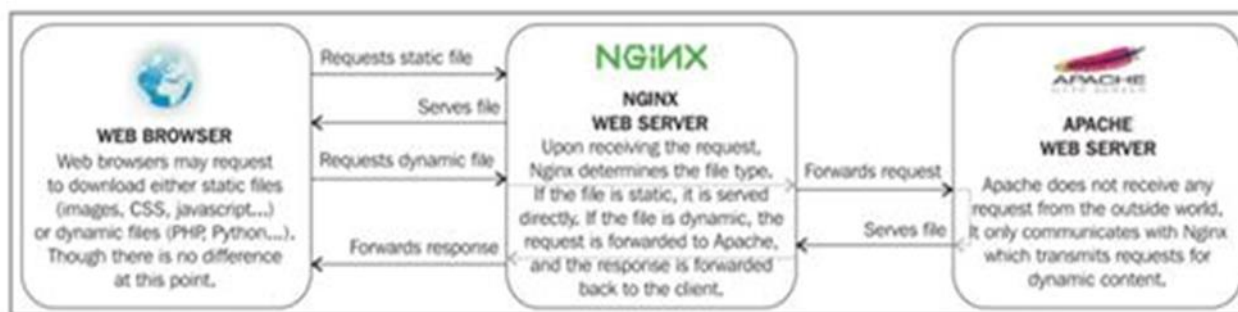
#### 2.1.4.1. Reverse proxy là gì?

Một proxy, theo định nghĩa, là một thiết bị đứng giữa server và client, tham gia vào "cuộc trò chuyện" giữa hai bên. Khái niệm proxy mà chúng ta thường dùng hàng

ngày được gọi là một forward proxy như chúng ta đã trình bày ở trên. Một reverse proxy làm công việc hoàn toàn ngược lại: nó đứng giữa một server và tất cả client mà server này phải phục vụ. Reverse proxy giống như một nhà ga kiêm một trạm kiểm soát, các request từ client, bắt buộc phải ghé vào reverse proxy, tại reverse proxy sẽ kiểm soát, lọc bỏ các request không hợp lệ, và luân chuyển các request hợp lệ đến đích cuối cùng là các server. Chú ý là một reverse proxy có thể luân chuyển request cho nhiều server cùng lúc. Lợi thế lớn nhất của việc sử dụng reverse proxy là ở khả năng quản lý tập trung. Một khi đã chuyển tất cả thông tin đi qua một trạm kiểm soát duy nhất (là reverse proxy), chúng ta có thể áp dụng nhiều biện pháp khác để tăng cường an ninh cho hệ thống của mình. Ngoài ra áp dụng reverse proxy đúng cách sẽ giúp tăng cường chất lượng cũng như nâng cao khả năng mở rộng của các ứng dụng web chạy trên các content server.

#### 2.1.4.2. Cơ chế hoạt động Reverse Proxy

Reverse Proxy là phương pháp dùng để giảm lưu lượng tải cho web server bằng sử dụng web cache giữ web server và internet. Đây còn là một trong những cách mở rộng hệ thống server mà không đòi hỏi độ phức tạp cao. Người ta thường dùng sử dụng reverse proxy để làm giảm gánh nặng cho webserver kể cả trang tĩnh và trang động. Hoạt động của reverse proxy được mô tả như hình sau:



Hình 5: Cơ chế hoạt động Reverse Proxy

- Cơ chế hoạt động của reverse proxy: Khi browser client tạo http request, sẽ được chuyển đến reverse proxy mà không trực tiếp đến webserver. Reverse proxy kiểm tra nội dung yêu cầu có nằm trong cache không, nếu không có sẽ connect trực tiếp đến webserver để download nội dung này về cache của nó và đồng thời trả kết quả về cho client. Reverse proxy cache các nội dung tĩnh (static) như js, image, doc, ... Còn nếu http request yêu cầu file nội dung động (dynamic) như: jsp, php, py, ... sẽ được forward xuống web server xử lý (do Nginx server xử lý các file dynamic không tốt bằng apache server).

## 2.1.4. LOAD BALANCING VỚI NGINX

### 2.1.4.1. Khái niệm load balancing

Cân bằng tải là một phương pháp phân phối khối lượng tải trên nhiều máy tính hoặc một cụm máy tính để có thể sử dụng tối ưu các nguồn lực, tối đa hóa thông lượng, giảm thời gian đáp ứng và tránh tình trạng quá tải trên máy chủ.

Các lợi ích khi sử dụng phương pháp cân bằng tải:

- Tăng khả năng đáp ứng, tránh tình trạng quá tải trên máy chủ, đảm bảo tính linh hoạt và mở rộng cho hệ thống.
- Tăng độ tin cậy và khả năng dự phòng cho hệ thống: Sử dụng cân bằng tải giúp tăng tính HA (High Availability) cho hệ thống, đồng thời đảm bảo cho người dùng không bị gián đoạn dịch vụ khi xảy ra lỗi sự cố lỗi tại một điểm cung cấp dịch vụ.
- Tăng tính bảo mật cho hệ thống. Thông thường khi người dùng gửi yêu cầu dịch vụ đến hệ thống, yêu cầu đó sẽ được xử lý trên bộ cân bằng tải, sau đó thành phần cân bằng tải mới chuyển tiếp các yêu cầu cho các máy chủ bên trong. Quá trình trả lời cho khách hàng cũng thông qua thành phần cân bằng tải, vì vậy mà người dùng không thể biết được chính xác các máy chủ bên trong cũng như phương pháp phân tải được sử dụng. Bằng cách này có thể ngăn chặn người dùng giao tiếp trực tiếp với các máy chủ, ẩn các thông tin và cấu trúc mạng nội bộ, ngăn ngừa các cuộc tấn công trên mạng hoặc các dịch vụ không liên quan đang hoạt động trên các cổng khác.

### 2.1.4.2. Các thuật toán load balancing của Nginx

Sau đây là một số thuật toán load balancing được nginx hỗ trợ:

- *Round-robin* là thuật toán phân bố đều số lượng request cho các webserver. Ví dụ chúng ta có hai webserver: request đầu tiên sẽ forward đến server 1, request thứ hai sẽ forward đến server 2, request thứ 3 thì forward lại server 1,....
- *Least\_conn* là thuật toán dựa trên số lượng kết nối của từng máy chủ, request sẽ gửi đến server có số lượng kết nối ít nhất.
- *Ip\_hash* là thuật toán được dựa trên địa chỉ IP của máy client. Lấy 3 octet đầu của địa chỉ IPv4 băm ra để xác định request từ client sẽ đến server nào. Như vậy, các request có cùng địa chỉ sẽ cùng sử dụng server nên session của website được giữ lại.

## 2.1.5. TOMCAT SERVER

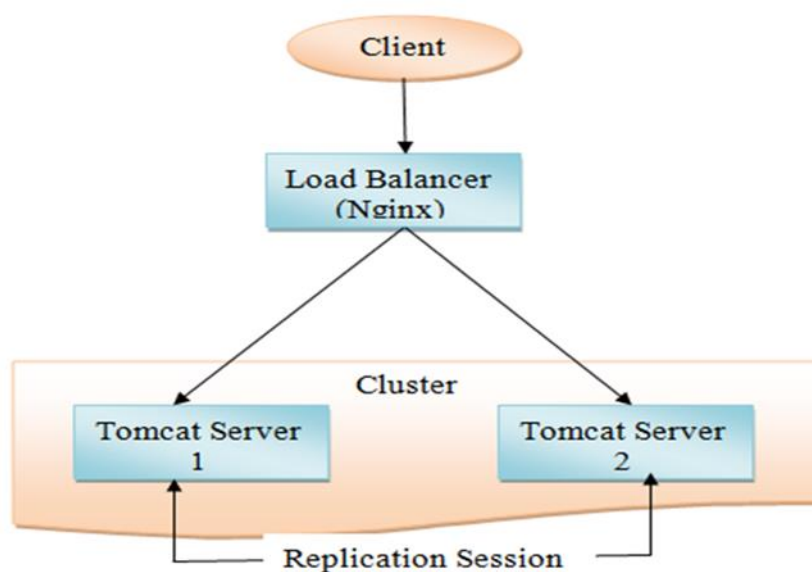
### 2.1.5.1. Giới thiệu Tomcat Server

Apache Tomcat là một Java Servlet được phát triển bởi Apache Software Foundation (ASF). Tomcat thi hành các ứng dụng Java Servlet và JavaServer Pages (JSP) từ Sun Microsystems, và cung cấp một máy chủ HTTP cho ngôn ngữ Java thuần túy để thực thi các chương trình lệnh viết bằng ngôn ngữ Java.

Tomcat không nên được hiểu nhầm với các máy chủ HTTP Apache - cái mà dùng để thực thi các câu lệnh viết bằng ngôn ngữ C trên máy chủ HTTP; có 2 máy chủ web được kết nối với nhau. Apache Tomcat cung cấp các công cụ cho việc cấu hình và quản lý, nhưng cũng có thể được cấu hình bởi việc soạn thảo các file cấu hình viết bằng XML.

### 2.1.5.2. Giới thiệu Session Replication

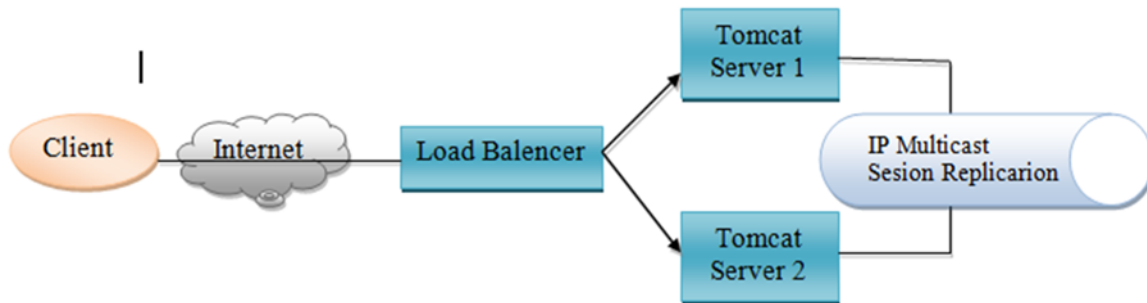
Session replication là hình thức trạng thái service nào đó được nhân bản (copy hay replication) qua nhiều application server instances khác. Session replication là chúng ta nhân bản trạng thái được lưu trữ trong HttpSession. Khi máy server phục vụ cho ta bị crashed, yêu cầu của ta sẽ được chuyển đến một server đang hoạt động khác trong hệ thống cluster, nhưng nếu session không được replicated trước đó, thì session của ta sẽ bị mất và dẫn đến lỗi. Đến đây ta thấy được sự cần thiết của chức session replication thật sự cần thiết cho môi trường clustering.



Hình 6: Replication Session

### 2.1.5.3. Cơ chế hoạt động replication session

Mô tả một yêu cầu Cluster Http Request được xử lý như thế nào.



Hình 7: Cơ chế hoạt động replication session

- Creation of session: Khi một session được khởi tạo, session sẽ được replicated.
- Attributes added to the session: giá trị session sẽ được replicate đến các nodes trong cluster.
- Attributes removed from the session: gỡ bỏ giá trị của session.
- Session expiration: nếu một session bị hết hạn trên một node, nó sẽ hết hạn trên tất cả các nodes trong cluster.
- Last access timestamp update: session sẽ không hết hạn trong các nodes khác trong cluster.
- Setting the user principal to the session: cho phép login state được replicate.
- Complete replication of all sessions: sử dụng một node mới khi join vào cluster.

Ví dụ:

- Ta có 2 tomcat server được cấu hình chạy trong một cluster. Gọi là 2 instance TC1 và TC2.
- Khi một HTTP request đến và load balancer chuyển cho TC1. Một session được tạo trong JSP/Servlet. Kết quả là TC1 sẽ broadcast một thông điệp đến tất cả các nodes trong cluster, trong trường hợp này là TC2. TC2 sẽ tạo một session giống như session trên TC1. Lúc này cả TC1 và TC2 đều có cùng một session trong bộ nhớ.
- JSP/Servlet thêm một attribute cho session, TC1 sẽ broadcast một thông điệp đến các nodes trong cluster. Lúc này bản thân của thông điệp được gửi chứa đựng attribute session và các server nhận thông điệp này sẽ bổ sung attribute vào session của nó.



- Khi một tomcat instance thứ 3 được startup, nó join vào hệ thống cluster và gửi thông điệp yêu cầu các danh sách session đang nắm giữ trên các nodes. Thông điệp này gọi là <get-all-sessions>, và các node sẽ response thông tin mà TC3 yêu cầu.

## **2.1.6. NoSQL**

### **2.1.6.1. Khái niệm NoSQL**

NoSQL thường được hiểu là Not Only SQL một dạng cơ sở dữ liệu cung cấp cơ chế lưu trữ và truy xuất dữ liệu theo mô hình khác với các cơ sở dữ liệu quan hệ. NoSQL được đánh giá là có phương thức tiếp cận thiết kế đơn giản, dễ dàng mở rộng ngang và có độ sẵn sàng đáp ứng cao, dễ dàng kiểm soát. Cấu trúc dữ liệu của NoSQL được lưu trữ dưới dạng: key-value, document hoặc graph khác với cách lưu trữ mà các RDBMS đang sử dụng hiện nay. Tính chất lưu trữ đơn giản, không ràng buộc vì vậy hiệu suất hoạt động của NoSQL nhanh hơn RDBMS rất nhiều.

NoSQL ra đời năm 1998 bởi Carlo Strozzi khi ông lập mới một hệ cơ sở dữ liệu quan hệ mã nguồn mở nhanh và nhẹ không liên quan đến SQL. Thuật ngữ NoSQL đánh dấu bước phát triển của thể hệ CSDL mới: phân tán (distributed) và không ràng buộc (non-relational).

### **2.1.6.2. Đặc điểm NoSQL**

- NoSQL lưu trữ dữ liệu của mình theo dạng cặp giá trị “key – value”. Sử dụng số lượng lớn các node để lưu trữ thông tin – Mô hình phân tán dưới sự kiểm soát phần mềm.
- Chấp nhận dữ liệu bị trùng lặp do một số node sẽ lưu cùng thông tin giống nhau.
- Một truy vấn sẽ được gửi tới nhiều máy cùng lúc, do đó khi một máy nào đó không phục vụ được sẽ không ảnh hưởng lắm đến chất lượng trả về kết quả.
- Phi quan hệ – không có ràng buộc nào cho việc nhất quán dữ liệu.
- Tính nhất quán không theo thời gian thực: Sau mỗi thay đổi CSDL, không cần tác động ngay đến tất cả các CSDL liên quan mà được lan truyền theo thời gian.

### **2.1.6.3. Các loại NoSQL**

- Key/values: Riak, Redis, Cassandra, Voldemort, Memcached.
- Column-Family: HBase, HyperTable, Cassandra.
- Document database: CouchDB, MongoDB.
- Graph database: Neo4j, OrientDB, InfiniteGraph, AllegroGraph.

Tuy cùng mang những đặc điểm chung nói trên của NoSQL nhưng một CSDL NoSQL cũng có những đặc điểm riêng, và vì thế thường được dùng cho những dự án khác nhau.

#### **2.1.6.4. So sánh giữa NoSQL và RDBMS**

Các RDBMs hiện tại đã bộc lộ những yếu kém như việc đánh chỉ mục một lượng lớn dữ liệu, phân trang, hoặc phân phối luồng dữ liệu media (phim, ảnh, nhạc...). Cơ sở dữ liệu quan hệ được thiết kế cho những mô hình dữ liệu nhỏ thường xuyên đọc viết trong khi các Social Network Services lại có một lượng dữ liệu cực lớn và cập nhật liên tục do số lượng người dùng quá nhiều ở một thời điểm. Thiết kế trên Distributed NoSQL giảm thiểu tối đa các phép tính toán, I/O liên quan kết hợp với batch processing đủ đảm bảo được yêu cầu xử lý dữ liệu của các mạng dịch vụ dữ liệu cộng đồng này. Facebook, Amazon là những ví dụ điển hình.

Về cơ bản, các thiết kế của NoSQL lựa chọn mô hình lưu trữ tập dữ liệu theo cặp giá trị keyvalue. Khái niệm node được sử dụng trong quản lý dữ liệu phân tán. Với các hệ thống phân tán, việc lưu trữ có chấp nhận trùng lặp dữ liệu. Một request truy vấn tới data có thể gửi tới nhiều máy cùng lúc, khi một máy nào đó bị chết cũng không ảnh hưởng nhiều tới toàn bộ hệ thống. Để đảm bảo tính real time trong các hệ thống xử lý lượng lớn, thông thường người ta sẽ tách biệt database ra làm 2 hoặc nhiều database. Một database nhỏ đảm bảo vào ra liên tục, khi đạt tới ngưỡng thời gian hoặc dung lượng, database nhỏ sẽ được gộp (merge) vào database lớn có thiết kế tối ưu cho phép đọc (read operation). Mô hình đó cho phép tăng cường hiệu suất I/O – một trong những nguyên nhân chính khiến performance trở nên kém.

Thế hệ CSDL mới – NoSQL – giảm thiểu tối đa các phép tính toán, tác vụ đọc-ghi liên quan kết hợp với xử lý theo lô (batch processing) đảm bảo được yêu cầu xử lý dữ liệu của các dịch vụ mạng xã hội. Hệ CSDL này có thể lưu trữ, xử lý từ lượng rất nhỏ đến hàng petabytes dữ liệu với khả năng chịu tải, chịu lỗi cao nhưng chỉ đòi hỏi về tài nguyên phần cứng thấp. Thiết kế đặc biệt tối ưu về hiệu suất, tác vụ đọc – ghi, ít đòi hỏi về phần cứng mạnh và đồng nhất, dễ dàng thêm bớt các node không ảnh hưởng tới toàn hệ thống.

Các mô hình dữ liệu đặc thù của NoSQL cung cấp API tự nhiên hơn so với việc dùng RDBMS.

Những ràng buộc về giấy phép sử dụng cùng với một khoản phí không nhỏ cũng là ưu thế. Chấp nhận NoSQL đồng nghĩa với việc bạn tham gia vào thế giới nguồn mở nơi mà bạn có khả năng tùy biến mạnh mẽ các sản phẩm, thư viện theo đúng mục đích của mình.



Tính năng	CSDL quan hệ	NoSQL
Hiệu suất	Kém hơn NoSQL	Cực tốt
	Relational giữa các table	Bỏ qua ràng buộc dữ liệu
Khả năng mở rộng	Hạn chế số lượng	Hỗ trợ lượng lớn các node
Hiệu suất đọc ghi	Kém do thiết kế để đảm bảo vào/ra liên tục của dữ liệu	Tối ưu đọc ghi dữ liệu

## 2.1.7. MONGODB

### 2.1.7.1. Giới thiệu MongoDB

MongoDB (được lấy tên từ “humongous”) là hệ thống CSDL mã nguồn mở dạng NoSQL. Thay vì lưu trữ cấu trúc dạng table như các Hệ CSDL quan hệ truyền thống, MongoDB lưu trữ dữ liệu dưới dạng JSON, khiến cho việc sử dụng nó trên một số loại ứng dụng trở nên dễ dàng và nhanh chóng hơn.

MongoDB đầu tiên được phát triển bởi công ty phần mềm 10gen (nay là MongoDB Inc) trong tháng 10 năm 2007 như một sản phẩm dịch vụ, công ty chuyển sang mô hình phát triển mã nguồn mở trong năm 2009, với việc 10gen cung cấp hỗ trợ thương mại và các dịch vụ khác. Kể từ đó, MongoDB đã được áp dụng trên rất nhiều hệ thống dịch vụ lớn, bao gồm Craigslist, eBay, Foursquare, SourceForge, Viacom, the New York Times, và nhiều hơn nữa. Mongo trở thành Hệ CSDL NoSQL phổ biến nhất.

MongoDB được viết bằng C++, và sau đây là những đặc điểm chính của MongoDB:

- Các truy vấn Ad hoc: Mongo hỗ trợ việc tìm theo trường, khoảng kết quả tìm và tìm theo cú pháp. Các truy vấn có thể trả về các trường được qui định trong văn bản và cũng có thể bao gồm các hàm Javascript mà người dùng chưa định nghĩa.
- Đánh chỉ mục: Bất cứ một trường nào trong MongoDB đều được đánh chỉ mục (giống như chỉ mục bên RMDBs).
- Mô phỏng (nhân bản): Mongo hỗ trợ mô phỏng Master-slave. Một master có thể điều khiển việc đọc và ghi. Một slave tạo bản sao dữ liệu từ master và chỉ được sử dụng cho việc đọc và backup (không có quyền ghi). Slave có khả năng chọn ra một master mới nếu master cũ bị hỏng.

- Cân bằng tải: Mongo mở rộng theo chiều ngang bằng cách sử dụng Sharding. Các lập trình viên chọn các khóa chia sẻ nhằm xác định dữ liệu sẽ được phân tán như thế nào. Dữ liệu sẽ được tách thành các khoảng dựa vào khóa và phân tán dọc theo các Shard.
- Lưu trữ file: Mongo lưu trữ bằng file hệ thống, rất tốt cho việc cân bằng tải và nhân bản dữ liệu. Trong các hệ thống nhiều máy, các file được phân phối và được sao ra rất nhiều lần giữa các máy một cách trong suốt. Do đó rất hiệu quả trong việc tạo ra một hệ thống cân bằng tải và dung lỗi tốt.

### 2.1.7.2. Kiến trúc tổng quát

Một MongoDB Server sẽ chứa nhiều database. Mỗi database lại chứa một hoặc nhiều collection. Đây là một tập các documents, về mặt logic thì chúng gần tương tự như các table trong CSDL quan hệ. Tuy nhiên, điểm hay ở đây là ta không cần phải định nghĩa trước cấu trúc của dữ liệu trước khi thao tác thêm, sửa dữ liệu... Một document là một đơn vị dữ liệu – một bản ghi (không lớn hơn 16MB). Mỗi chúng lại chứa một tập các trường hoặc các cặp key – value. Key là một chuỗi ký tự, dùng để truy xuất giá trị dạng: string, integer, double... Dưới đây là một ví dụ về MongoDB document:

```
{
  "_id" : "1111428",
  "_class" : "edu.ctu.elcit.model.Student",
  "password" : "huunhan",
  "hoten" : "Đinh Hữu Nhân",
  "diachi" : "Đồng Tháp",
  "gioitinh" : "Nam",
  "ngaysinh" : "13/11/1993",
  "lop" : "DI11Y9A1",
  "khoaahoc" : 37,
  "manganh" : "Y9",
  "nganhhoc" : "Truyền thông và mạng máy tính",
  "khoa" : "Công nghệ thông tin và truyền thông"
}
```

Hình 8: Cấu trúc một document

Cấu trúc có vẻ khá giống JSON, tuy nhiên, khi lưu trữ document này ra database, MongoDB sẽ serialize dữ liệu thành một dạng mã hóa nhị phân đặc biệt – BSON. Ưu điểm của BSON là hiệu quả hơn các dạng format trung gian như XML hay JSON cả hệ tiêu thụ bộ nhớ lẫn hiệu năng xử lý. BSON hỗ trợ toàn bộ dạng dữ liệu mà JSON hỗ trợ (string, integer, double, Boolean, array, object, null) và thêm một số dạng dữ liệu đặc biệt như regular expression, object ID, date, binary, code.



Hình 9: So sánh RDBMS với MongoDB

### 2.1.7.3. Index

Chỉ mục làm tăng hiệu suất truy vấn lên rất nhiều. Điều quan trọng là nghĩ xem xét tất cả các loại truy vấn cần trong ứng dụng để xác định những chỉ mục liên quan. Khi đã xác định xong, việc tạo ra các chỉ mục trong MongoDB là khá dễ dàng.

Tham khảo chi tiết phần 2 phụ lục A.

### 2.1.7.4. Truy vấn

Giống như CSDL quan hệ, MongoDB cũng hỗ trợ truy vấn với các câu điều kiện phức tạp. Robomongo là một công cụ cho phép thiết lập kết nối và thực hiện các truy vấn cũng như hiển thị kết quả với các câu truy vấn MongoDB, hình dưới là giao diện tổng quát của Robomongo.

Tham khảo chi tiết phần 3 phụ lục A.

### 2.1.8. REPLICATION MONGODB

Có lẽ công việc quan trọng nhất của bất kỳ quản trị viên MongoDB là đảm bảo sao cho sao chép được thiết lập và hoạt động đúng. Sao chép có thể được sử dụng hoàn toàn để dự phòng và toàn vẹn dữ liệu hoặc có thể được sử dụng cho mục đích cao hơn như mở rộng đọc, sao lưu nóng,...

MongoDB hỗ trợ sao chép dữ liệu không đồng bộ giữa các máy chủ. Tại một thời điểm, chỉ có 1 máy chủ hoạt động để ghi (primary hay master).

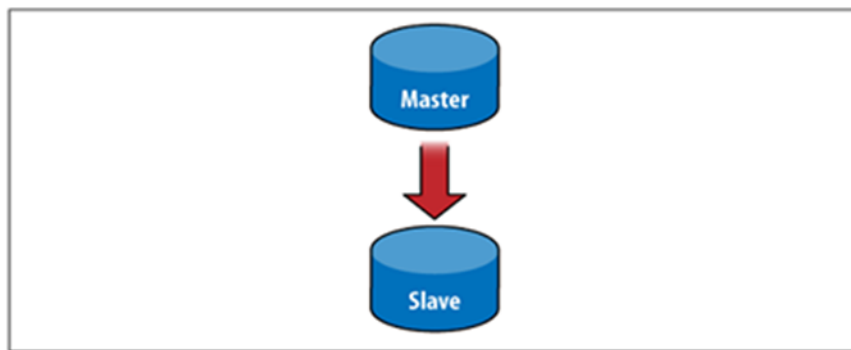
Có hai hình thức sao chép.

- Master-Slave Replication.
- Replica Sets.

#### 2.1.8.1. Master-Slave Replication

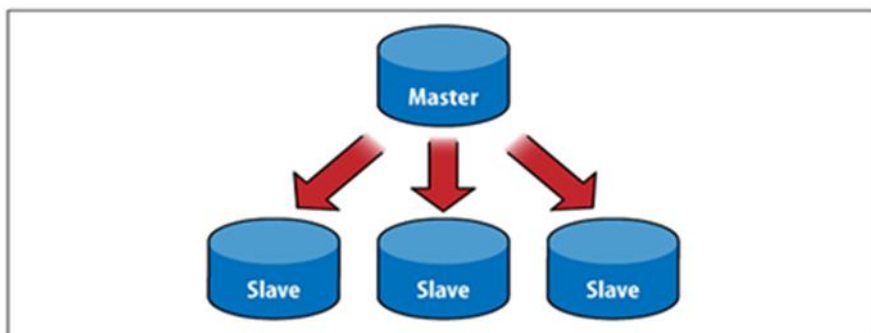
Sao chép Master-slave là mô hình sao chép phổ biến nhất được hỗ trợ bởi MongoDB. Mô hình này rất linh hoạt và có thể được sử dụng để sao lưu, dự phòng, mở rộng đọc, ...

Hình 10 minh họa mô hình Master – Slave bao gồm 2 nút, một nút làm Master, nút còn lại làm Slave



*Hình 10: Mô hình Master – Slave hai nút*

Hình 11 minh họa mô hình Master – Slave bao gồm 4 nút, một nút làm Master, 3 nút còn lại làm Slave



*Hình 11: Mô hình Master – Slave bốn nút*

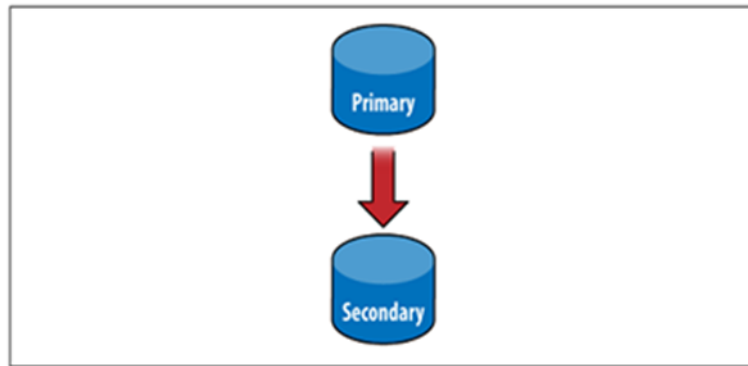
Để thiết lập cần khởi động nút master và một hoặc nhiều nút slave, các nút này đều biết địa chỉ của nút master. Để khởi động master, chạy mongod --master. Để khởi

động slave, chạy mongod --slave --source master\_address, trong đó master\_address là địa chỉ của nút master vừa được khởi động.

### 2.1.8.2. Replica Set

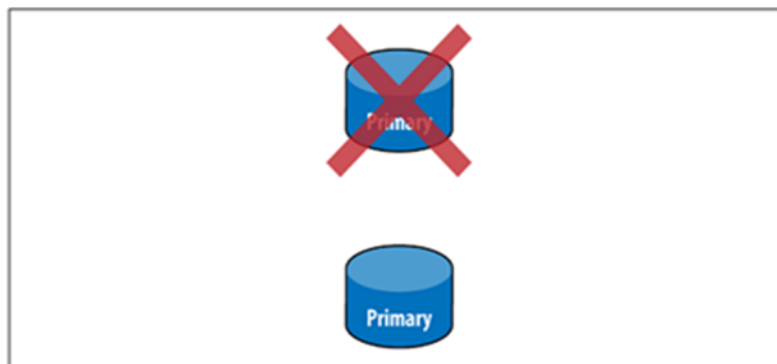
Replica Sets là một cụm master-slave tự động chịu lỗi. Replica Sets không có một master cố định: một master được bầu chọn và có thể thay đổi đến nút khác nếu master bị sập.

Hình 12 mô phỏng mô hình Replica Sets gồm 2 nút.



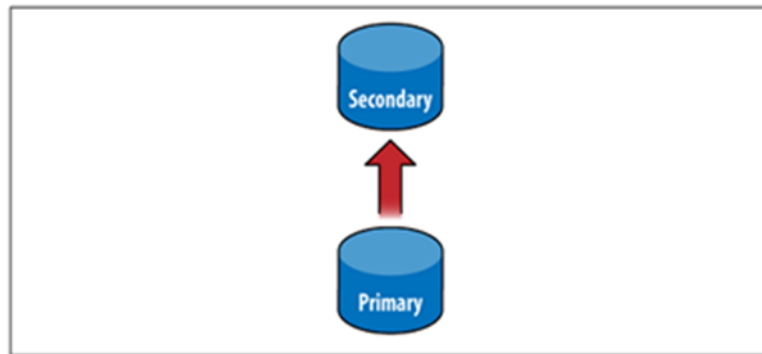
*Hình 12: Mô hình Replica Sets hai nút*

Khi server chính chết, server cấp 2 chờ thành server chính (hình 11).



*Hình 13: Replica Sets – Bầu chọn master mới*

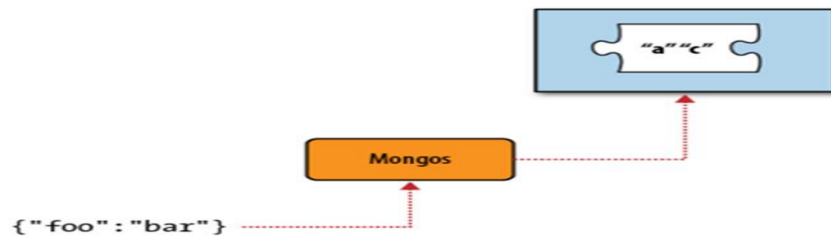
Nếu server chính ban đầu hoạt động trở lại, nó trở thành server cấp 2 (hình 13).



Hình 14: Server chính trở thành server cấp 2

### 2.1.9. SHARDING MONGODB

Sharding là cơ chế tự động của MongoDB dùng để chia tách một dữ liệu kích thước lớn cho rất nhiều server (thường gọi là cluster). Sharding được thiết kế để phục vụ 3 mục điều cơ bản sau:

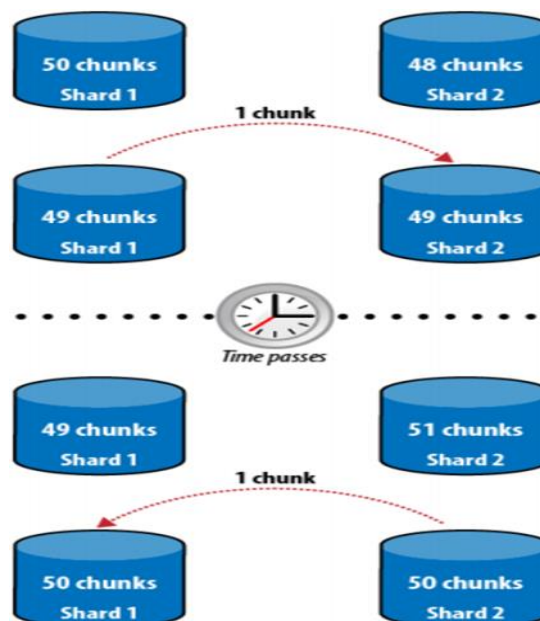


Hình 15: Cơ chế Sharding

- Làm cho cluster “trong suốt” với người dùng: Để hoàn thành nhiệm vụ này, MongoDB sử dụng một quá trình routing đặc biệt gọi là mongos. Mongos đứng trước cluster, đóng vai trò điều phối việc truy cập đến shard nào, trả dữ liệu từ shard nào ra. Nó chuyển tiếp các request tới các server khác có tài nguyên hoặc đến cluster đằng sau nó. Sau đó lắp ráp lại, và gửi các response lại về cho các client. Do đó, các client không cần biết rằng chúng đang giao tiếp với cluster nào thật sự mà chỉ biết rằng mình đang kết nối tới một server bình thường. Đây gọi là tính “trong suốt” với người sử dụng.
- Làm cho cluster luôn sẵn sàng để đọc hoặc ghi: Một cluster còn tồn tại phải đảm bảo được rằng nó luôn sẵn sàng. Mỗi phần con trong cluster sẽ có ít nhất một vài tiến trình phục vụ dự bị trên máy khác.
- Làm cho cluster phát triển “tự nhiên”: Bất cứ khi nào người dùng cần thêm dung lượng, họ có thể thêm một cách dễ dàng. Mỗi cluster khi được quản lý lại “thể hiện” như một node riêng lẻ và dễ dàng config.

### 2.1.9.1. Balancing – Cân bằng tải

Nếu có nhiều shard đang sẵn sàng và có thêm chứa thêm dữ liệu, MongoDB sẽ tiến hành chuyển dữ liệu từ các shard khác sang để cân bằng tải. Cách thức tiến hành là di chuyển các chunk từ shard này sang shard khác một cách tự động. Balancing cũng có thể bị tắt hoặc bật nếu admin muốn Balancing cũng không được đảm bảo ngay tức thì, chúng ta hãy xem ví dụ dưới đây:



Hình 16: Balancing MongoDB

### 2.1.10. CÔNG CỤ APACHE JMETER

#### 2.1.10.1. Apache JMeter là gì?

Apache JMeter là một phần mềm nguồn mở được viết bằng Java nhằm mục đích kiểm thử chức năng và hiệu suất. Mục đích ban đầu JMeter được thiết kế chỉ để kiểm thử các ứng dụng web nhưng hiện nay nó đã được mở rộng thêm nhiều chức năng khác.

Cha đẻ của JMeter là Stefano Mazzocchi, một lập trình viên tại Apache Software Foundation. Ông ta viết JMeter với mục đích là kiểm thử hiệu năng của Apache JServ (bây giờ là Apache Tomcat). Sau đó Apache đã thiết kế lại để cải tiến hơn giao diện đồ họa cho người dùng và khả năng kiểm thử hướng chức năng.

Nó là một ứng dụng Java với phân giao diện sử dụng Java Swing, do đó nó có thể chạy được trên mọi nền tảng có hỗ trợ JVM, ví dụ như Windows, Linux, Mac,... Các tính năng nổi bật của JMeter:



- Nguồn mở, miễn phí
- Giao diện đơn giản, trực quan dễ sử dụng
- Có thể kiểm thử nhiều kiểu server: Web - HTTP, HTTPS, SOAP, Database - JDBC, LDAP, JMS, Mail - POP3,...
- Một công cụ độc lập có thể chạy trên nhiều nền tảng hệ điều hành khác nhau, trên Linux chỉ cần chạy bằng một shell scrip, trên Windows thì chỉ cần chạy một file .bat
- JMeter lưu các kịch bản kiểm thử của nó dưới dạng các file XML, do đó ta có thể tự tạo các kịch bản kiểm thử của mình bằng một trình soạn thảo bất kỳ và load nó lên
- Đa luồng, giúp xử lý tạo nhiều request cùng một khoảng thời gian, xử lý các dữ liệu thu được một cách hiệu quả
- Đặc tính mở rộng, có rất nhiều plugin được chia sẻ rộng rãi và miễn phí
- Một công cụ tự động để kiểm thử hiệu năng và tính năng của ứng dụng.
- Cách thức hoạt động: nó giả lập một nhóm người dùng gửi các yêu cầu tới một máy chủ mục tiêu, nhận và xử lý các response từ máy chủ và trình diễn các kết quả đó cho người dùng dưới dạng bảng biểu, đồ thị,...

#### **2.1.10.2. Phạm vi ứng dụng của JMeter**

Cùng chức năng Load testing còn có rất nhiều công cụ khác như NeoLoad, LoadRunner, Appvance, ... Trong đó thì công cụ LoadRunner là nổi tiếng hơn hẳn, nhưng so với JMeter thì nó còn có một số hạn chế như sau:

- Chỉ sử dụng được trên Windows
- Không miễn phí
- Chỉ hỗ trợ giao thức nền HTTP
- JMeter thì nổi trội hơn do hỗ trợ rất nhiều các giao thức như:
- Web: HTTP, HTTPS sites 'web 1.0' web 2.0 (ajax, flex and flex-ws-amf)
- Web Services: SOAP / XML-RPC
- Database: JDBC
- Directory: LDAP
- Messaging Oriented service: JMS
- Service: POP3(s), IMAP(s), SMTP(s)
- TCP
- MongoDB (NoSQL)



## 2.2. CÀI ĐẶT GIẢI PHÁP

### 2.2.1. MÔI TRƯỜNG CÀI ĐẶT CLUSTER

#### 2.2.1.1. Hạ tầng phần cứng

Hạ tầng phần cứng bao gồm 6 server và một máy Client có cấu hình phần cứng với các thông số cơ bản như sau:

- **Nginx server** (có chức năng load balancer, reverse proxy):
  - CPU: 1 core.
  - Ram: 512 MB.
  - HDD 20GB
- **Hai Tomcat Server :**
  - CPU: 1 core.
  - Ram: 1GB.
  - HDD 20GB
- **MongoDB Server** (chức năng Index):
  - CPU: 1 core.
  - Ram: 521 MB.
  - HDD 20GB
- **Hai MongoDB Server** (chức năng Shard):
  - CPU: 1 core.
  - Ram: 1GB.
  - HDD 40GB.
- **Máy client chạy Jmeter** ( dùng để kiểm thử)

#### 2.2.1.2. Hạ tầng phần mềm

Các dịch vụ cơ bản được cài đặt trên hệ thống đáp ứng yêu cầu cơ bản của web server (các server được chạy trên máy ảo Virtuabox):

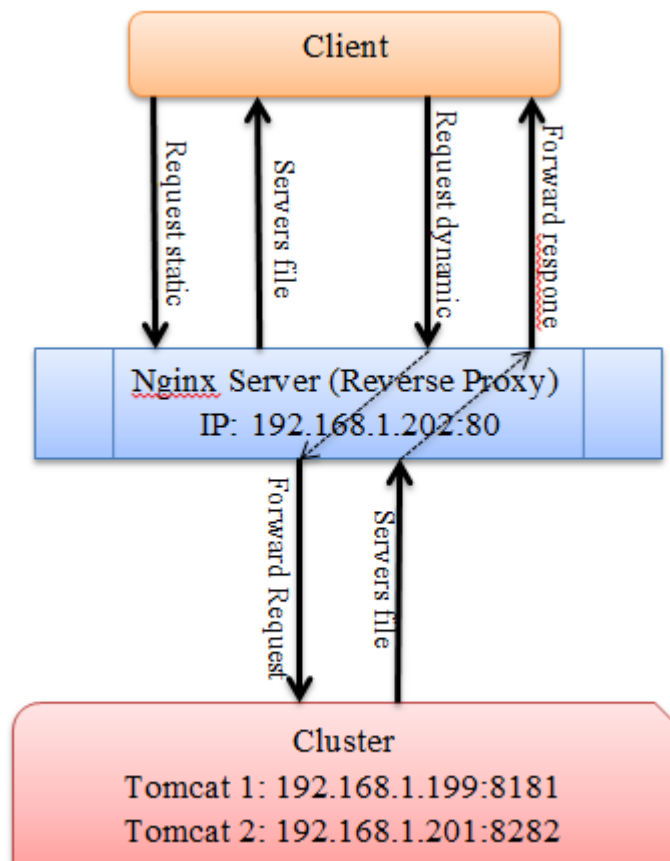
Server	Nginx	Tomcat 1	Tomcat 2	MongoDB (Index)	MongoDB (Shard) 1	MongoDB (Shard) 1	Jmeter
Operating system	Ubuntu server 14.04						Ubuntu Desktop 14.04
Tomcat server		✓	✓				

<b>Nginx</b>	✓						
<b>MongoDB</b>				✓	✓	✓	
<b>JDK, Apache Jmeter</b>							✓

Bảng 4: Cơ sở hạ tầng phần mềm

## 2.2.2. CẤU HÌNH REVERSE PROXY

### 2.2.2.1. Mô hình triển khai



Hình 17: Mô hình triển khai reverse proxy

### 2.2.2.2. Cấu hình reverse proxy

Để cấu hình reverse proxy cho tomcat server hoặc cluster tomcat server, chúng ta tiến hành cài nginx server:

- Cài đặt Nginx Server (Tham khảo phụ lục).
- Chỉnh sửa file cấu hình của Nginx : */etc/nginx/nginx.conf*
- Tạo một virtual server bằng cách thêm vào khối http

```
server {  
    listen    80;  
    server_name 192.168.1.202;  
    location  
~*^.+.(jpg/jpeg/gif/png/ico/css/zip/tgz/gz/rar/bz2/doc/xls/exe/pdf/ppt/txt/tar/mid/midi/w  
av/bmp/rtf/js)$ {  
        root /var/static;  
        expires -1;  
        access_log off;  
        add_header Cache-Control "public";  
    }  
}
```

Giải thích các thông số:

- Nginx server lắng nghe cổng 80.
- Server\_name là 192.168.1.202 (có thể đặt tên miền).
- *location*  
~\*^.+.(jpg/jpeg/gif/png/ico/css/zip/tgz/gz/rar/bz2/doc/xls/exe/pdf/ppt/txt/tar/mid/  
midi/wav/bmp/rtf/js)\$ dùng để lọc các yêu cầu qua nếu những gói tin là một  
trong số các định dạng trên sẽ được nginx cache lại chứa vào thư mục  
/var/static.

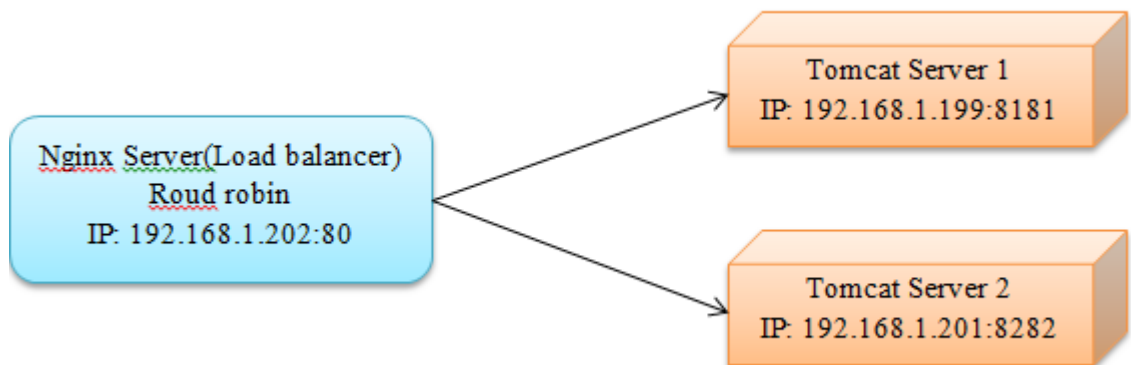
### 2.2.3. LOAD BALANCING VỚI NGINX

#### 2.2.3.1. Mô hình triển khai load balancing với nginx

Các thành phần trong mô hình (tất cả các server được triển khai trên Ubuntu Server 14.04):

- **Tomcat Server 1:**
  - IP: 192.168.1.199
  - Port: 8181

- **Tomcat Server 2:**
  - IP: 192.168.1.201
  - Port: 8282
- **Nginx Server:**
  - IP: 192.168.1.202
  - Port: 80



Hình 18: Mô hình triển khai load balancing

#### 2.2.3.2. Cấu hình load balancing với nginx

- Cài đặt dịch vụ webserver nginx trên Ubuntu server.
- Sửa thông tin file `/etc/nginx/nginx.conf`
  - Tạo một cluster gồm 2 Tomcat server có IP và port như trên. Thêm vào khối http

```
upstream loadbalancer{
    #ip_hash;
    # least_conn;
    server 192.168.1.199:8181;
    server 192.168.1.201:8282;
}

○ Tạo Virtual Host trong khối http:
server {
    listen    80;
    server_name 192.168.1.202;
    location / {
        proxy_pass http://loadbalancer/;
    }
}
```

- Khởi động lại nginx server.

Hướng dẫn cài đặt chi tiết: tham khảo phần 2 phụ lục B

## 2.2.4. CẤU HÌNH REPLICATION SESSION VỚI TOMCAT SERVER

Mô hình cài đặt Replication Session Tomcat Server:



Hình 19: Mô hình cài đặt Session Replication

Một số giá trị quan trọng Replication Session:

- Địa chỉ multicast là 228.0.0.4
- Cổng multicast là 45564
- Địa IP broadcasted là `java.net.InetAddress.getLocalHost().getHostAddress()`  
(Lưu ý: địa chỉ 127.0.0.1 không thể broadcast nên chúng ta phải thay đổi địa chỉ 127.0.0.1 thành địa chỉ IP của Tomcat Server trong file: `/etc/hosts`)
- Các cổng TCP lắng nghe replication thông điệp sẽ trong khoảng: 4000-4100
- Cả hai tomcat server đều cấu hình: `ClusterSessionListener`

Sau đây là cấu hình mặc định của các tomcat server trong cluster trong file:

`Tomcat/conf/server.xml`:

```
<Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"
  channelSendOptions="8">

  <Manager className="org.apache.catalina.ha.session.DeltaManager"
    expireSessionsOnShutdown="false"
    notifyListenersOnReplication="true"/>

  <Channel className="org.apache.catalina.tribes.group.GroupChannel">
    <Membership className="org.apache.catalina.tribes.membership.McastService"
      address="228.0.0.4"
      port="45564"
      frequency="500"
      dropTime="3000"/>
    <Receiver className="org.apache.catalina.tribes.transport.nio.NioReceiver"
      address="auto"
      port="4000"
      autoBind="100"
      selectorTimeout="5000"
      maxThreads="6"/>

    <Sender className="org.apache.catalina.tribes.transport.ReplicationTransmitter">
      <Transport className="org.apache.catalina.tribes.transport.nio.PooledParallelSender"/>
    </Sender>
    <Interceptor className="org.apache.catalina.tribes.group.interceptors.TcpFailureDetector"/>
    <Interceptor className="org.apache.catalina.tribes.group.interceptors.MessageDispatch15Interceptor"/>
  </Channel>

  <Valve className="org.apache.catalina.ha.tcp.ReplicationValve"
    filter=""/>
  <Valve className="org.apache.catalina.ha.session.JvmRouteBinderValve"/>

  <Deployer className="org.apache.catalina.ha.deploy.FarmWarDeployer"
    tempDir="/tmp/war-temp/"
    deployDir="/tmp/war-deploy/"
    watchDir="/tmp/war-listen/"
    watchEnabled="false"/>

  <ClusterListener className="org.apache.catalina.ha.session.JvmRouteSessionIDBinderListener">
  <ClusterListener className="org.apache.catalina.ha.session.ClusterSessionListener">
</Cluster>
```

Hướng dẫn cấu hình chi tiết replication session: tham khảo phụ lục

## 2.2.5. SHARDING MONGODB

MongoDB mở rộng theo chiều ngang bằng cách sử dụng sharding. Các lập trình viên chọn các khóa chia sẻ nhằm xác định dữ liệu sẽ được phân tán như thế nào. Dữ liệu sẽ được tách thành các khoảng dựa vào khóa và phân tán dọc theo các Shard.

### 2.2.5.1. Một số khái niệm trong việc sharding MongoDB

- **Shard:** là server MongoDB dùng để lưu trữ một số document của một collection.
- **Mongos:** có chức năng định tuyến (routing) các request từ client. Những thông tin định tuyến được lấy từ config server.
- **Config server:** dùng để lưu trữ các cấu hình của cluster. Ví dụ như: vị trí của document nằm trong shard nào, nó sẽ là thông tin để cho mongos định tuyến.

### 2.2.5.2. Thông tin cấu hình cluster MongoDB

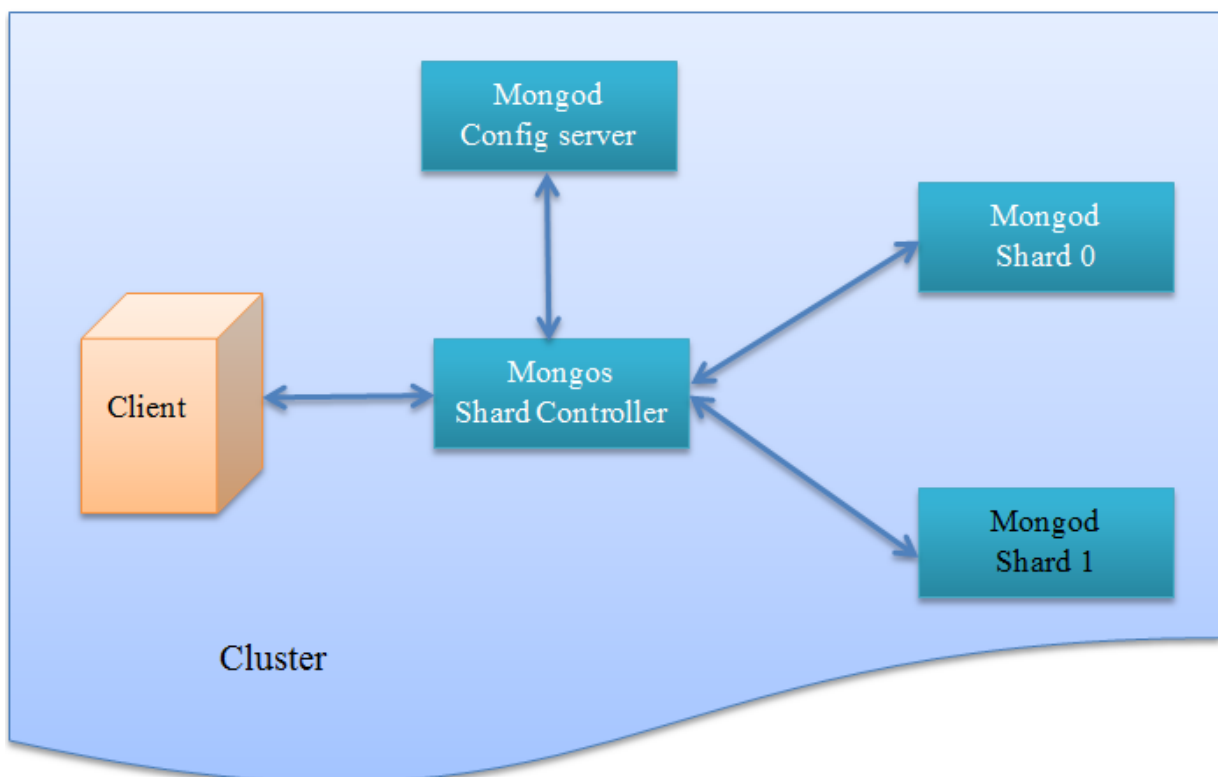
Mô hình Cluster gồm 3 server:

- Mongo 1: dùng làm config và routing server.
- Mongo 2 và mongo3: làm shard server.

Service	Daemon	IP	Port	dbpath
Shard controller (routing)	mongos	192.168.1.222	27017	/data/db/
Config server	mongod	192.168.1.222	27018	/data/config/
Shard 0	mongod	192.168.1.223	27017	/data/db/
Shard 1	mongod	192.168.1.224	27017	/data/db/

Bảng 5: Cấu hình cluster MongoDB

### 2.2.5.3. Mô hình triển khai cluster MongoDB



Hình 20: Mô hình triển khai MongoDB

## CHƯƠNG 3: KIỂM THỬ VÀ ĐÁNH GIÁ

### 3.1. MỤC TIÊU KIỂM THỬ

Từ mục tiêu trên, để đánh giá kiểm thử được performance của hệ thống cluster, ta dựa vào các yếu tố sau đây:

- Request rate: số lượng connection mà hệ thống có khả năng đáp ứng đồng thời, do đó hệ thống có performance cao là hệ thống có khả năng đáp ứng đồng thời tối đa connection đến nhiều. Để đo giá trị performance “thực” của hệ thống thông cluster cần phải xây dựng một hệ thống phân tán các clients. Khi đó ta mới xác định giá trị ngưỡng (giá trị tới hạn) số request đồng thời mà hệ thống có khả năng đáp ứng.
- Throughput hệ thống: số byte tối đa mà hệ thống truyền qua trong một đơn vị thời gian (một giây). Điều này phụ thuộc vào request rate, độ trễ trung bình giữa các request và số byte trung bình của mỗi request.
- Kiểm tra xem hệ thống có chịu tải được bao nhiêu người dùng cùng một lúc, sự khác nhau khi chạy một server tomcat và nhiều server tomcat.
- Quan sát CPU, RAM của từng server trong cluster.

### 3.2. KỊCH BẢN KIỂM THỬ

#### 3.2.1. Kịch bản mô phỏng đăng ký học phần

Địa chỉ của website: <http://192.168.1.202/>

Bước	Đường dẫn	Dữ liệu	Phương thức	Kết quả
Trang đăng nhập hệ thống	/		GET	Trang yêu cầu đăng nhập hệ thống
Nhập mssv và password và nhấn nút đăng nhập	/checklogin	Mssv = \${mssv} Password = \${pass}	POST	Nếu thành công sẽ thấy được trang thông tin sinh viên, ngược lại thất bại nhận thông báo sai mật khẩu



Vào trang đăng ký học phần	/dkhp		GET	Nhận được trang đăng ký học phần
Xem các lớp học phần của mã học phần CT333	/xem/CT333		GET	Nhận được danh sách lớp học phần của mã học phần CT333
Đăng ký học phần CT333 của mã lớp học phần CT33301	/dkhocphan/CT333/CT33301		POST	Nhận được thông báo đăng ký thành công hay thất bại (do hết lớp học phần hoặc trùng thời khóa biểu)
Xem các lớp học phần của mã học phần CT337	/xem/CT337		GET	Nhận được danh sách lớp học phần của mã học phần CT337
Đăng ký học phần CT337 của mã lớp học phần CT33701	/dkhocphan/CT333/CT33701		POST	Nhận được thông báo đăng ký thành công hay thất bại (do hết lớp học phần hoặc trùng thời khóa biểu)
Xem các lớp học phần của mã học phần CT338	/xem/CT338		GET	Nhận được danh sách lớp học phần của mã học phần CT338
Đăng ký học phần CT333 của mã lớp học phần CT33801	/dkhocphan/CT338/CT33801		POST	Nhận được thông báo đăng ký thành công hay thất bại (do hết lớp học phần)

				hoặc trùng thời khóa biểu)
Xem các lớp học phần của mã học phần CT307	/xem/CT307		GET	Nhận được danh sách lớp học phần của mã học phần CT307
Đăng ký học phần CT333 của mã lớp học phần CT30701	/dkhocphan/CT307/CT30701		POST	Nhận được thông báo đăng ký thành công hay thất bại (do hết lớp học phần hoặc trùng thời khóa biểu)
Xem các lớp học phần của mã học phần CT329	/xem/CT329		GET	Nhận được danh sách lớp học phần của mã học phần CT329
Đăng ký học phần CT329 của mã lớp học phần CT32901	/dkhocphan/CT333/CT32901		POST	Nhận được thông báo đăng ký thành công hay thất bại (do hết lớp học phần hoặc trùng thời khóa biểu)
Xem các lớp học phần của mã học phần CT336	/xem/CT336		GET	Nhận được danh sách lớp học phần của mã học phần CT336
Đăng ký học phần CT336 của mã lớp học phần CT33601	/dkhocphan/CT336/CT33601		POST	Nhận được thông báo đăng ký thành công hay thất bại (do hết lớp học phần hoặc trùng thời

				khóa biểu)
Xóa học phần CT333 có mã học phần CT33301	/xoa/CT33301/CT333		POST	Xóa đăng ký sẽ nhận được thông báo thành công.
Xem thay đổi mã học phần CT329	/thaydoi/CT329/CT32901		GET	Danh sách lớp học phần của mã học phần CT329
Chọn thay đổi của mã học phần CT329 của mã lớp học phần CT32901	/thaydoilop/CT329/CT32902/CT32901		GET	Nếu thay đổi sẽ nhận được thông báo thành công, ngược lại nhận được thông báo thất bại (do hết lớp học phần hoặc trùng thời khóa biểu)
Xem thời khóa biểu	/thoikhoabieu		GET	Xem thời khóa biểu của những môn vừa đăng ký.
Đăng xuất	/logout		GET	Nhận được trang đăng nhập

Bảng 6: Kịch bản kiểm thử

#### Diễn giải kịch bản:

- **Trang đăng nhập hệ thống:** khi sinh viên thực hiện đăng ký học phần sẽ vào trang đăng nhập được cung cấp địa chỉ cho trước, địa chỉ là <http://192.168.1.202/>
- **Sinh viên đăng nhập hệ thống:** sinh viên trường cung cấp tài khoản trước, sau đó nhập mã số sinh viên và password. Click nút đăng nhập, nếu thành công chúng ta sẽ nhận được giao diện thông tin sinh viên, ngược lại thất bại nhận được thông báo sinh mã số sinh viên hoặc password.

- **Trang đăng ký học phần:** Sau khi đăng nhập thành công tiếp theo chúng ta vào trang đăng ký học phần: <http://192.168.1.202/dkhp>.
- **Xem các lớp học phần của mã học phần CT333:** sinh viên muốn xem danh sách lớp học phần của mã học phần CT333, sinh viên chọn vào radio button trên màn hình, nếu học phần có mở lớp sẽ nhận được danh sách lớp học phần, ngược lại lớp học phần không mở lớp sẽ nhận được thông báo lớp học phần không mở lớp. (Các môn khác xem tương tự).
- **Đăng ký học phần CT333 của mã lớp học phần CT33301 :** sau khi xem lớp học phần ta xem danh sách lớp học phần, chọn một lớp học phần đăng ký bằng cách nhấn nút “Đăng ký”. Nếu đăng ký thành công sẽ nhận được đăng ký thành công, ngược lại nhận thông báo đăng ký thất bại do trùng thời khóa biểu hoặc đã đủ sĩ số (các học phần khác tương tự).
- **Xóa học phần CT333 có mã học phần CT33301:** sau khi đăng ký thành công chúng ta sẽ nhận được kết quả sau khi đăng ký, nếu bạn muốn xóa học phần nào trong danh sách học phần đã đăng ký bằng chọn nút “Xóa” (các học phần khác tương tự).
- **Xem thay đổi mã học phần CT329 và chọn mã lớp học phần:** sau khi đăng ký thành công chúng ta sẽ nhận được kết quả sau khi đăng ký. Chọn nút “thay đổi” sẽ hiện lên một form có chứa danh sách các lớp học phần và chọn lớp học phần cần thay đổi nếu thành công sẽ nhận được thông báo thành công, nếu thất bại do hết sĩ số hoặc trùng thời khóa biểu (các học phần tương tự)

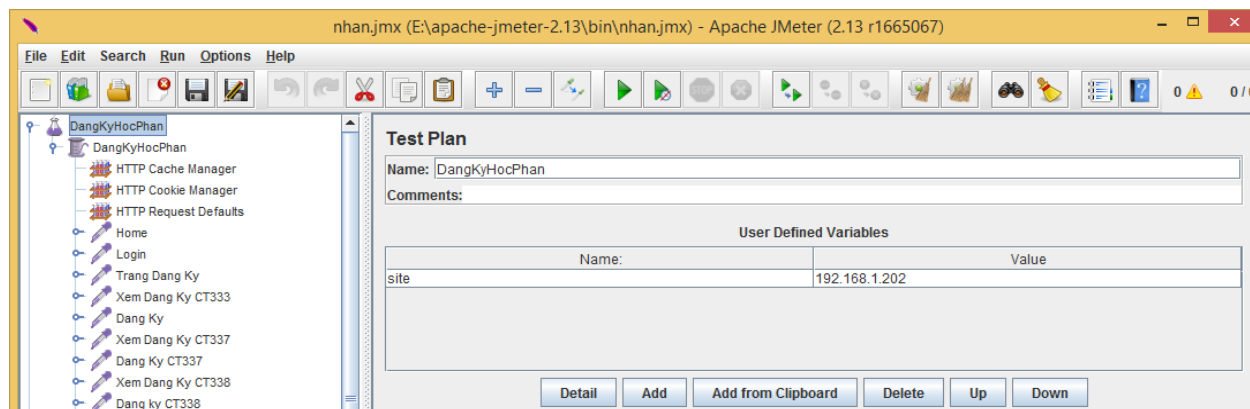
### 3.2.2. Cài đặt Apache Jmeter theo kịch bản

Kịch bản mô phỏng gồm một test plan, được đặt tên “DangKyHocPhan”. Do yêu cầu của kịch bản là phải tạo ra nhiều người dùng khác nhau truy cập nên trong test plan cần thêm vào 3 thành phần “HTTP Cache Manager”, “HTTP Cookie Manager” và “HTTP Request Default” giúp Jmeter giống như một trình duyệt web thật sự có thể lưu những giá trị cookie cũng như session của từng user. Trong test plan bao gồm nhiều “http request”, mỗi “http request” tương ứng với một thao tác của người dùng trong khi thi, cần tạo ra khoảng thời gian thực hiện của mỗi yêu cầu “http request” thông qua thành phần “Uniform Random Timer”, thành phần “Synchronizing timer” dùng để tạo ra sự đồng bộ cho một số yêu cầu.

### 3.2.2.1 Các thành phần chung

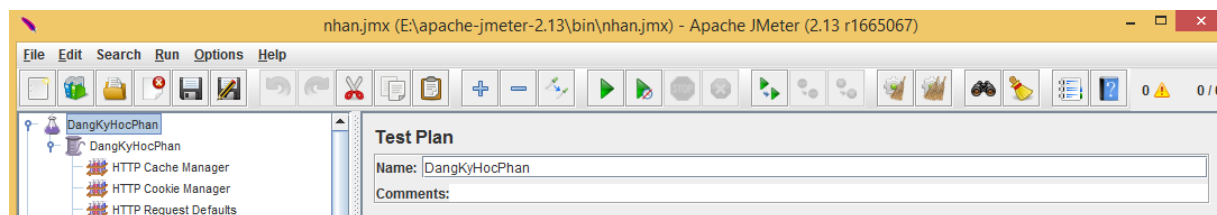
**Test Plan:** tạo test plan có tên “DangKyHocPhan” để chứa các thành phần khác nhau: Thread Group, Sampler và Listener.

Đặt giá trị biến site là <http://192.168.1.202/>



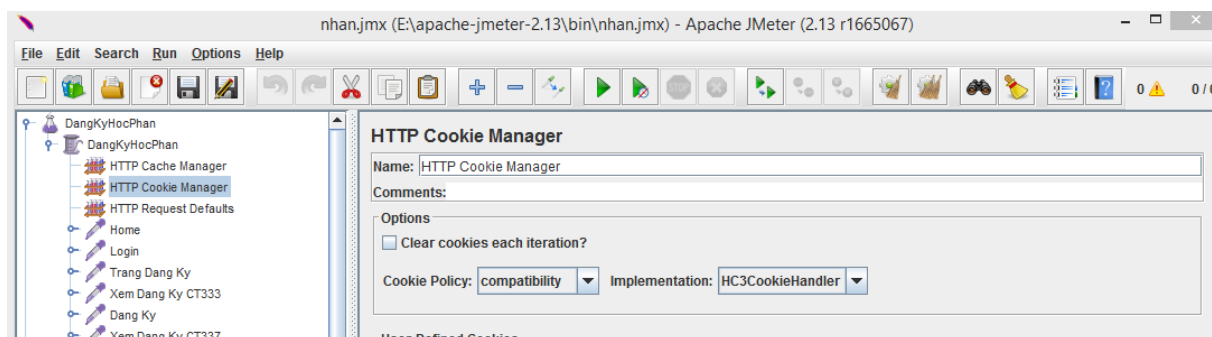
Hình 21: Test Plan

**Thread group:** trong test plant vừa tạo, thêm vào một thread group có tên “DangKyHocPhan”.



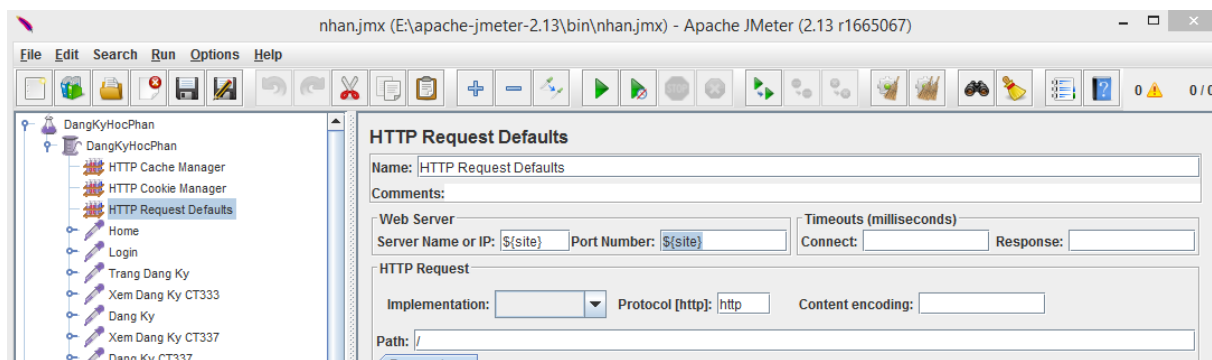
Hình 22: Thread group

Thêm thành phần HTTP Cache Manager, HTTP Cookie Manager nhằm giúp Jmeter có tính năng quản lý phiên làm việc như là một trình duyệt thực thụ, nếu các HTTP request và HTTP response có chứa Cookie thì Jmeter sẽ lưu lại sử dụng cho các trang khác nhau trong một Thread/User hay nói cách khác mỗi sinh viên thi trực tuyến sẽ được cấp một cookie được lưu trữ riêng.



Hình 23: HTTP Cookie Manager

Thêm thành phần HTTP Request Defaults để thêm các thông tin về tên, địa chỉ, cổng dịch vụ máy chủ website được sử dụng trong quá trình đánh giá. Với địa chỉ của server là biến  $\$ \{site\}$  (được khai báo trên Test Plan).



Hình 24: HTTP Request Defaults

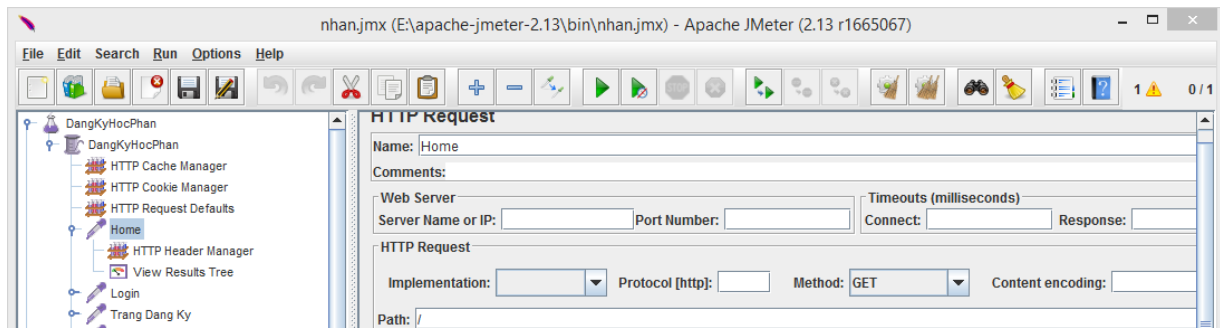
### 3.2.2.2. Tùy chỉnh Http Request

Thêm một Http request vào Thread Group bằng cách: Click chuột phải Thread Group → Add → Chọn Sampler → Http Request.

Thêm một HTTP Header Manager vào HTTP Request: Click chuột phải vào HTTP Request → Add → chọn Config Element → HTTP Header Manager.

Xem kết quả trả về của HTTP Request → Click chuột vào HTTP Request → Add → Listener → View Result Tree.

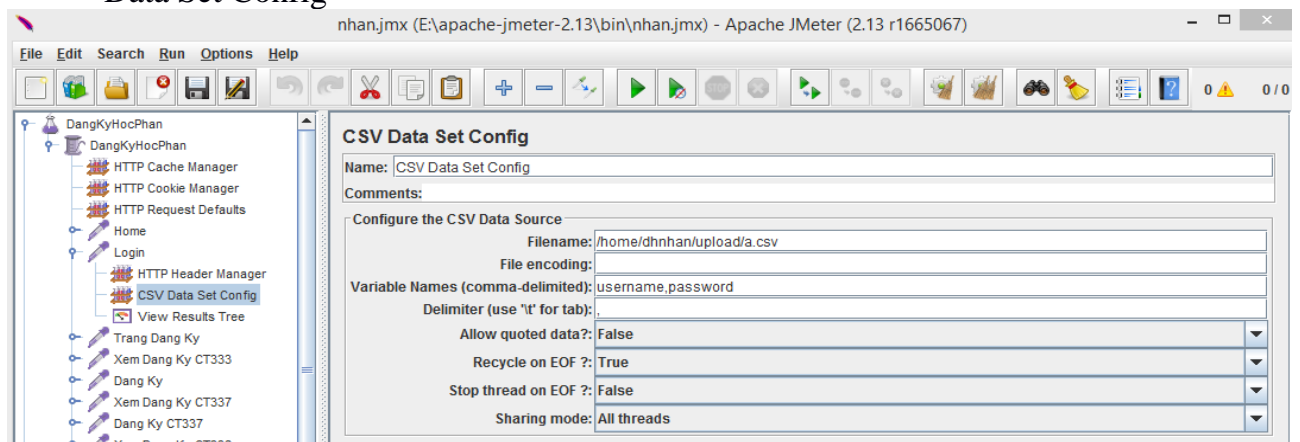
Tùy chỉnh Http Request Home: có path là /, method: GET, không cần đặt chỉ IP vào port do đã khai báo trên HTTP Request Defaults.



Hình 25: HTTP Request Home

Tùy chỉnh HTTP Request Login:

- Path: /checklogin
- Method: POST
- Thêm vào HTTP Request: CSV Data set Config để login với nhiều sinh viên khác nhau: click chuột phải HTTP Request → Add → Config Element → CSV Data Set Config

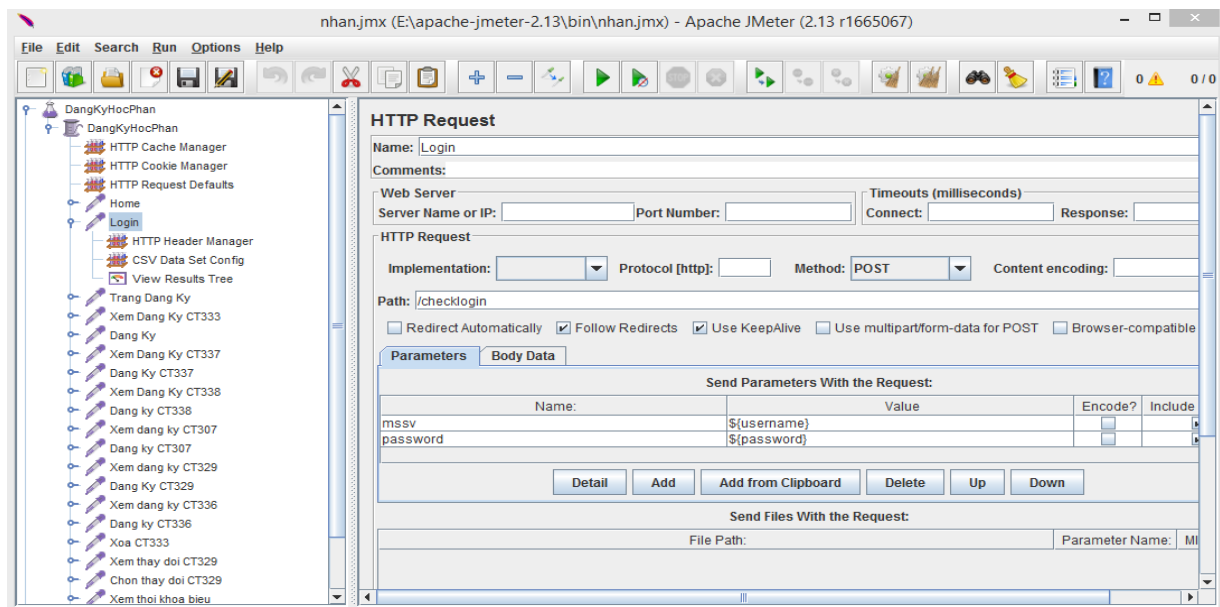


Hình 26: CSV Data set Config

Cấu trúc file CSV:

```
1111428,huunhan
1111429,huunhan
1111430,huunhan
1111431,huunhan
1111432,huunhan
1111433,huunhan
1111434,huunhan
1111435,huunhan
1111436,huunhan
```

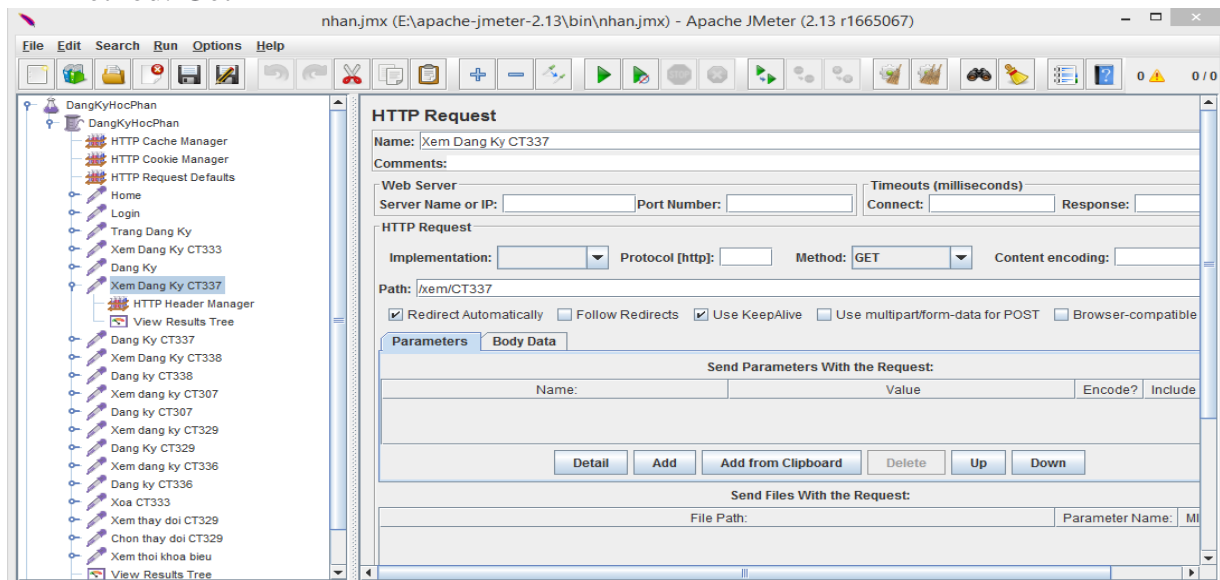
- Http Request:



Hình 27: HTTP Request login

Tùy chỉnh HTTP Request xem đăng ký mã học phần CT333 (các học phần khác tương tự)

- Path: /xem/CT337
- Method: Get

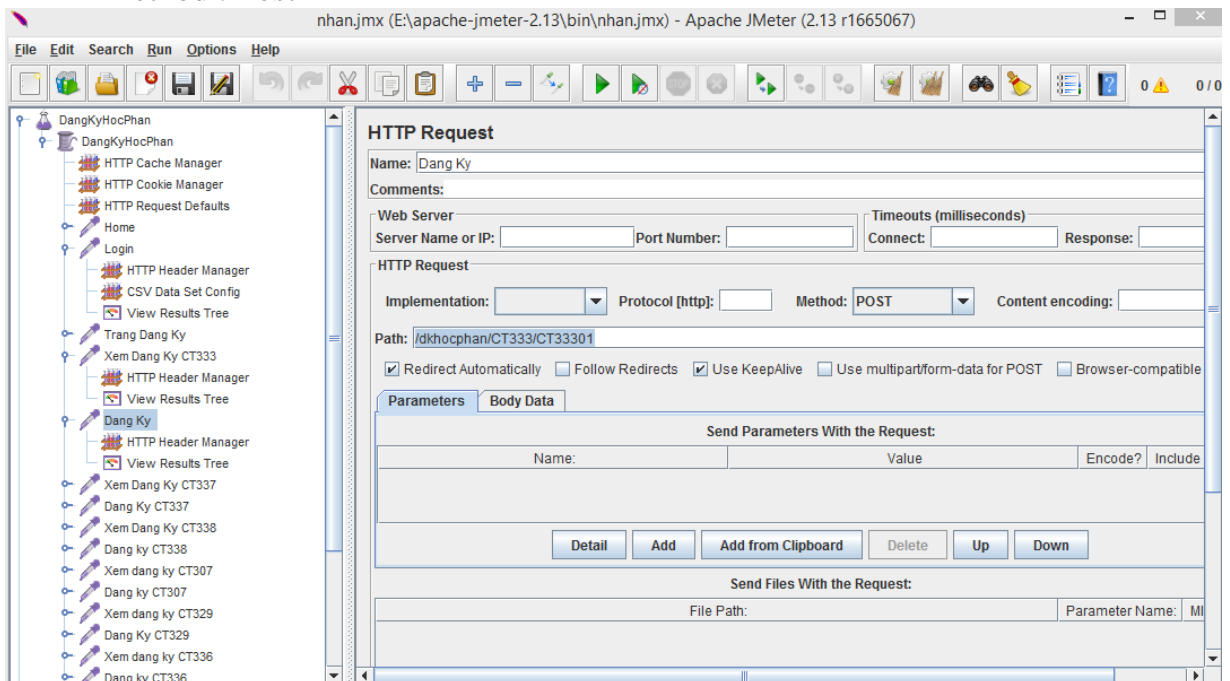


Hình 28: HTTP Request xem đăng ký mã học phần CT333

Tùy chỉnh HTTP Request đăng ký mã học phần CT333 (các học phần khác tương tự):



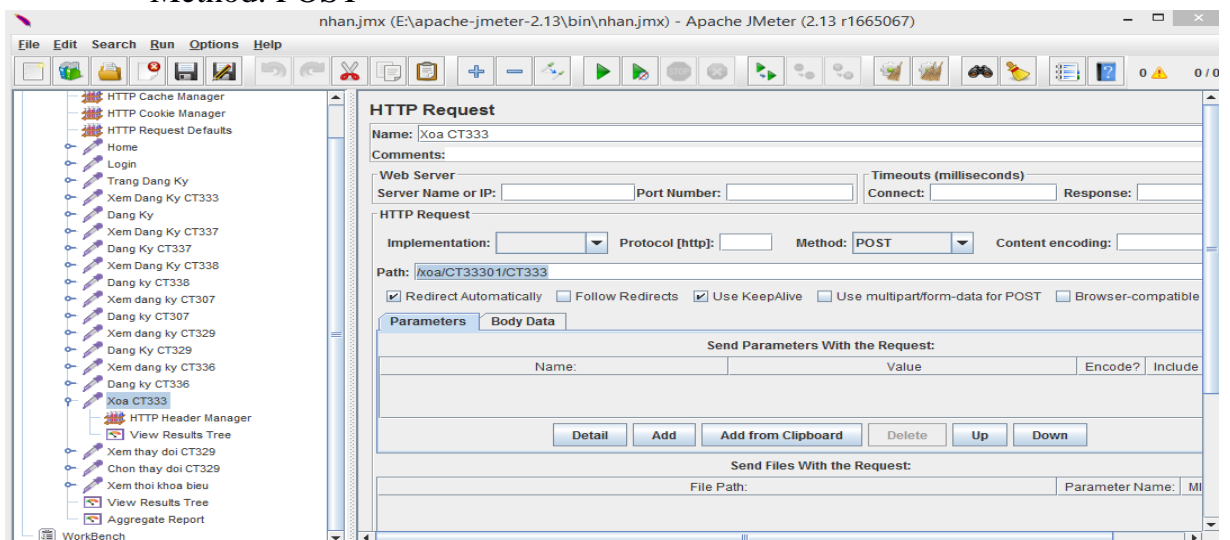
- Path: /dkhocphan/CT333/CT33301
- Method : Post



Hình 29: HTTP Request đăng ký mã học phần CT333

Tùy chỉnh HTTP Request xóa mã học phần CT333 (các học phần khác tương tự):

- Path: /xoa/CT33301/CT333
- Method: POST



Hình 30: HTTP Request xóa mã học phần CT333

### 3.3. KẾT QUẢ KIỂM THỬ

#### 3.3.1. Quá trình thực hiện

##### 3.3.1.1. Đánh giá kết quả mô phỏng 400 user với một Tomcat Server

Kết quả thông tin các server trong cluster khi chỉ sử dụng 1 tomcat server.

Server	Tên đại lượng tải	Giá trị tải
Mongos (Routing )	CPU	15 → 19 %
	RAM	124/490 MB
Mongod Shard 0	CPU	11 → 15%
	RAM	129/490
Mongod Shard 1	CPU	16 → 20%
	RAM	128/490 MB
Nginx server	CPU	15 → 20 %
	RAM	115/490MB
Tomcat Server 1	CPU	100%
	RAM	412/994 MB

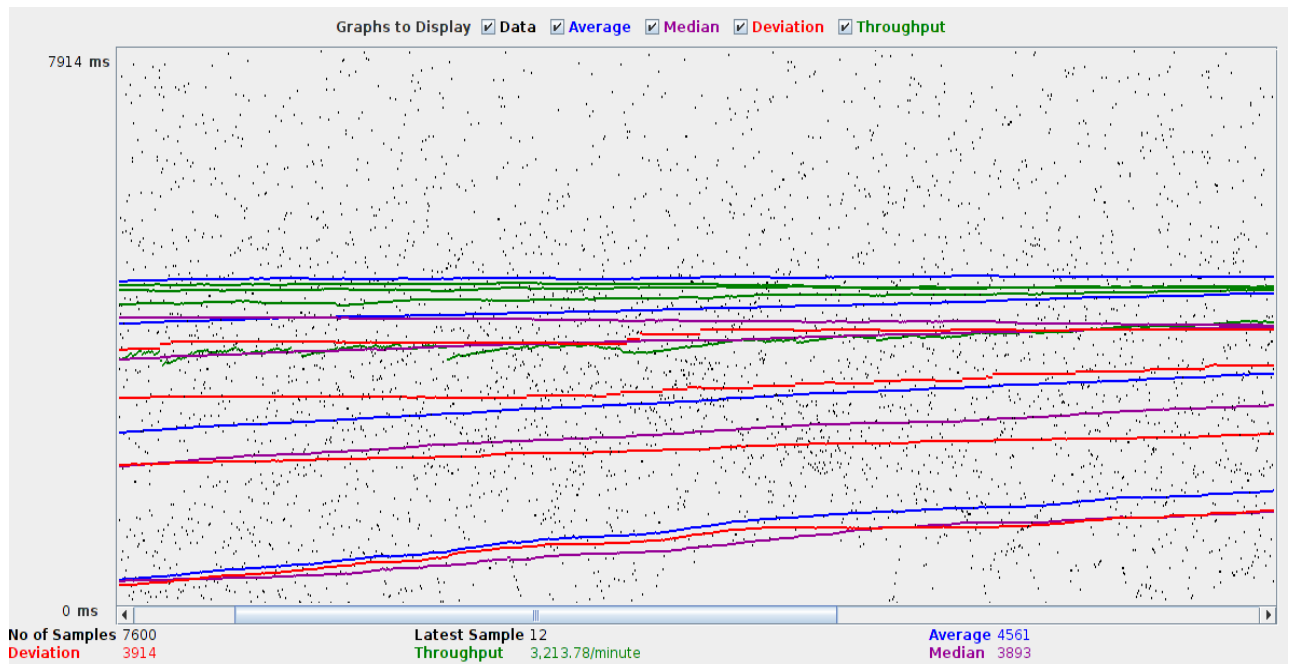
*Bảng 7: Thông tin server sử dụng một Tomcat server*

Sau đây là một số kết quả thu được từ công cụ Jmeter:

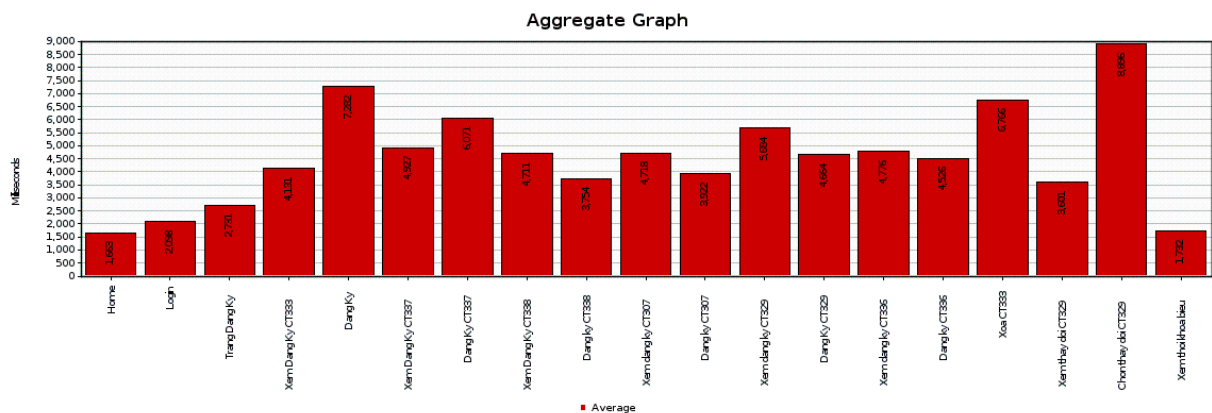
Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	KB/sec
Home	400	1663	1181	2903	4130	16715	7	34932	0.00%	4.3/sec	14.1
Login	400	2098	1701	3881	4633	6488	13	36189	0.00%	4.1/sec	22.8
Trang Da...	400	2731	2000	5149	6260	9612	24	66288	0.00%	3.5/sec	33.3
Xem Dan...	400	4131	3700	6510	7891	21043	103	37390	0.00%	3.3/sec	35.6
Dang Ky ...	400	7282	7172	11492	13256	25473	327	43441	0.00%	3.1/sec	.6
Xem Dan...	400	4927	4486	7397	8488	20651	121	67707	0.00%	3.0/sec	34.4
Dang Ky ...	400	6071	5525	9794	10891	17344	330	38013	0.00%	3.0/sec	.6
Xem Dan...	400	4711	4337	7102	8262	13299	276	66144	0.00%	3.0/sec	35.0
Dang ky ...	400	3754	3495	5771	6482	10840	178	34150	0.00%	3.0/sec	.6
Xem dan...	400	4718	4423	6994	8252	17756	404	34712	0.00%	2.9/sec	34.3
Dang ky ...	400	3922	3521	5717	6519	17742	582	36902	0.00%	2.9/sec	.6
Xem dan...	400	5684	5143	8323	9449	20826	501	38613	0.00%	2.9/sec	37.8
Dang Ky ...	400	4664	4118	7686	8641	11234	176	33371	0.00%	3.0/sec	.6
Xem dan...	400	4776	4376	7172	8621	12605	83	36022	0.00%	3.0/sec	35.6
Dang ky ...	400	4526	3822	7896	9409	12672	132	19060	0.00%	3.0/sec	.6
Xoa CT333	400	6766	6804	9421	10109	12053	192	17705	0.00%	3.0/sec	.6
Xem thay...	400	3601	3209	5866	6883	9883	101	66504	0.00%	3.1/sec	7.0
Chon tha...	400	8896	9465	13746	14682	17098	46	19264	0.00%	3.2/sec	5.4
Xem thoi ...	400	1732	1252	3453	4205	8415	12	32582	0.00%	3.2/sec	20.7
TOTAL	7600	4561	3893	8436	10389	15805	7	67707	0.00%	53.6/sec	283.9

*Bảng 8: Kết quả thu được sử dụng một Tomcat server*

Bảng kết quả trả về của kịch bản 400 user cùng lúc sử dụng một tomcat server



Biểu đồ 1: kết quả 400 user cùng lúc sử dụng một tomcat server



Biểu đồ 2: Thời gian xử lý request 400 user cùng lúc sử dụng một Tomcat server

### Nhận xét:

- Thông qua số liệu trên ta có thể nói cluster hiện đang bị quá tải ở Tomcat server (do hiệu suất của CPU đã tăng ngưỡng 100%).
- Thời gian xử lý các request khá lâu.
- Các Server mongoDB hoặc động bình thường. Việc đọc ghi dữ liệu nhanh đủ để đáp ứng cho tomcat server cần, Dung lượng CPU và RAM không ở mức quá tải.

- Số lượng byte thông qua là 283.9KB/sec và tổng throughput là 53.6/sec là tương đối nhỏ.

**Kết luận:** Mô hình cluster khi sử dụng một tomcat server đang bị quá tải ở server Tomcat mức CPU đạt tới mức tối đa là 100%. Thời gian xử lý request của cluster tương đối dài. Nhưng các server Database phục vụ tương đối tốt. Vậy khi cluster khi sử dụng một tomcat server là không thể đáp ứng 400 user cùng một lúc.

### 3.3.1.2. Đánh giá kết quả mô phỏng 400 user với hai Tomcat Server

Kết quả thông tin các server trong cluster khi chỉ sử dụng hai tomcat server.

Server	Tên đại lượng tải	Giá trị tải
Mongos (Routing )	CPU	25 → 31%
	RAM	133/490 MB
Mongod Shard 0	CPU	20 → 28 %
	RAM	129/490
Mongod Shard 1	CPU	15 → 24%
	RAM	156/490 MB
Nginx server	CPU	15 → 20 %
	RAM	127/490MB
Tomcat Server 1	CPU	40 → 75 %
	RAM	353/994MB
Tomcat server 2	CPU	50 → 73 %
	RAM	372/994

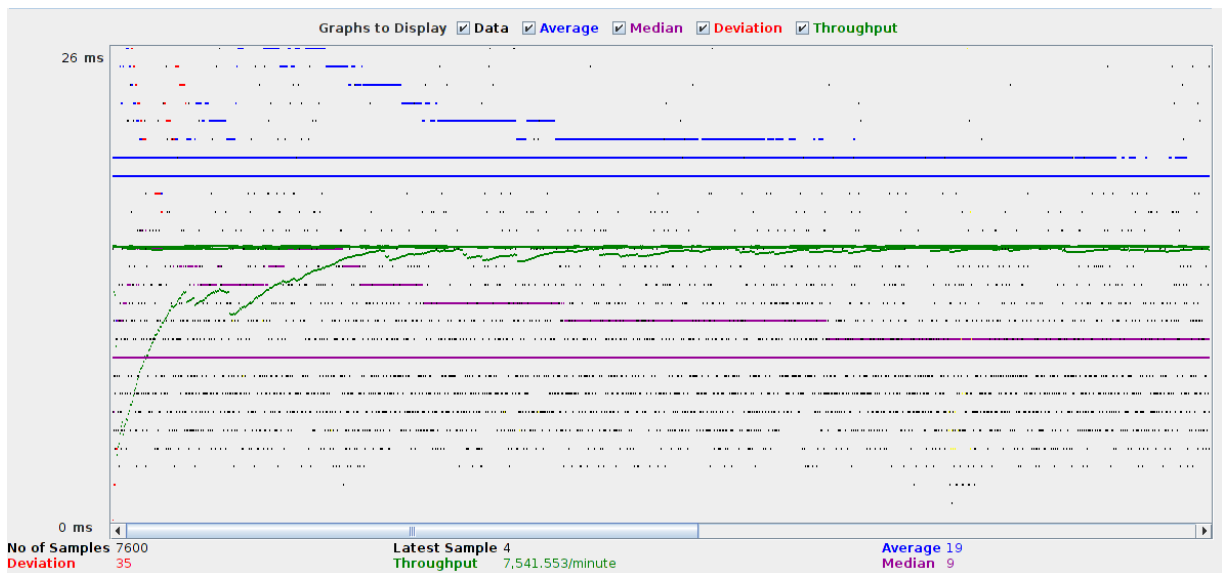
Bảng 9: Bảng thông tin server khi sử dụng 2 tomcat server

Sau đây là một số kết quả thu được từ công cụ Jmeter:

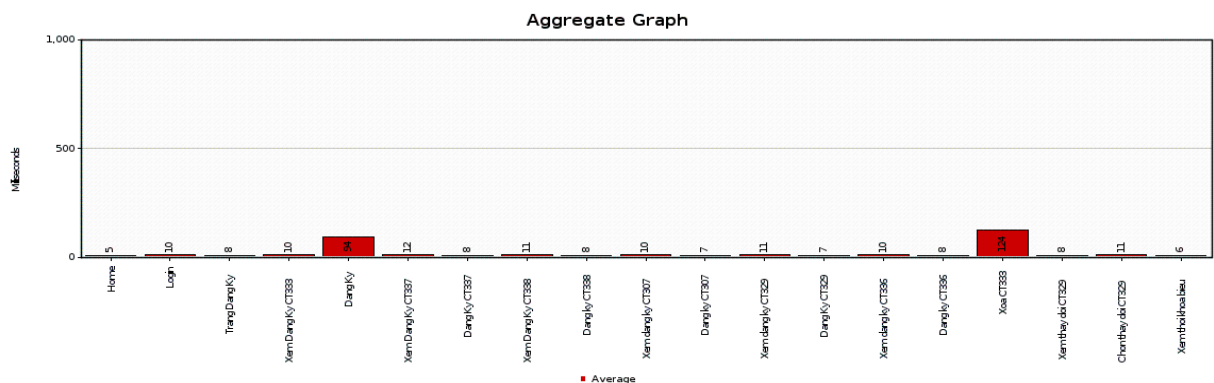
## Phát triển đăng ký môn học hướng thông lượng người dùng lớn

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	KB/sec
Home	400	5	5	8	10	15	2	37	0.00%	6.6/sec	21.9
Login	400	10	10	16	19	24	5	35	0.00%	6.6/sec	36.8
Trang Dang...	400	8	8	12	13	20	4	51	0.00%	6.6/sec	72.3
Xem Dang K...	400	10	10	16	18	22	5	55	2.25%	6.6/sec	78.6
Dang Ky	400	94	102	159	174	269	1	334	0.00%	6.6/sec	1.6
Xem Dang K...	400	12	11	17	20	26	4	54	1.25%	6.7/sec	82.1
Dang ky CT3...	400	8	7	13	15	23	1	36	0.00%	6.7/sec	1.5
Xem Dang K...	400	11	11	17	20	30	4	55	1.50%	6.7/sec	81.9
Dang ky CT3...	400	8	8	12	15	17	2	32	0.00%	6.7/sec	1.5
Xem dang k...	400	10	9	16	19	25	5	34	1.50%	6.7/sec	82.7
Dang ky CT3...	400	7	7	12	14	20	2	24	0.00%	6.7/sec	1.5
Xem dang k...	400	11	10	16	19	23	4	47	1.50%	6.7/sec	87.7
Dang Ky CT3...	400	7	7	11	15	20	1	28	0.00%	6.7/sec	1.4
Xem dang k...	400	10	10	17	20	25	4	31	1.25%	6.7/sec	82.5
Dang ky CT3...	400	8	7	13	16	23	2	28	0.00%	6.7/sec	1.4
Xoa CT333	400	124	120	171	199	287	2	317	0.00%	6.7/sec	1.4
Xem thay do...	400	8	8	12	14	23	3	27	1.50%	6.7/sec	15.3
Chon thay d...	400	11	11	17	20	26	5	31	1.50%	6.7/sec	11.5
Xem thoi kh...	400	6	5	9	11	17	2	42	0.00%	6.7/sec	43.0
TOTAL	7600	19	9	24	115	167	1	334	0.64%	125.7/sec	701.9

Bảng 10: Kết quả trả về của kịch bản 400 user cùng lúc sử dụng hai tomcat server



Biểu đồ 3: Kết quả 400 user cùng lúc sử dụng hai tomcat server



Biểu đồ 4: Thời gian xử lý request 400 user cùng lúc sử dụng hai Tomcat server

**Nhận xét:**

- Thông qua những số liệu về server và tool jmeter thì cluster không bị quá tải. Có thể đáp ứng được 400 user cùng một lúc.
- Hiệu suất CPU và RAM của các server trong cluster tương đối ổn định. Đối với các server database thì tương đối giống khi mô phỏng một tomcat server.
- Thời gian xử lý các request rất nhỏ so với cluster sử dụng một tomcat server (so sánh cụ thể hai biểu đồ giữa một tomcat server và hai tomcat server).
- Số lượng byte thông qua là 701.9 KB/sec hơn rất nhiều so với sử dụng một Tomcat server (283,9 KB/sec).
- Tổng Throughput của các request là 125,7/sec cũng lớn hơn nhiều so với sử dụng một tomcat server (53,6/sec).
- Có tổng khoảng 0.64% request bị error, lỗi này là do việc replication session không kịp thời.

**Kết luận:** Khi sử dụng hai tomcat server trong cluster thì việc quá tải không xảy ra nữa, chịu được một lượng tải cao hơn so với sử dụng một tomcat server. Có khả năng đáp ứng 400 user cùng một lúc mà server không bị quá tải.

**3.3.1.3. Đánh giá kết quả mô phỏng 800 user với hai Tomcat Server**

Kết quả thông tin các server trong cluster khi chỉ sử dụng hai tomcat server khi 800 user cùng lúc.

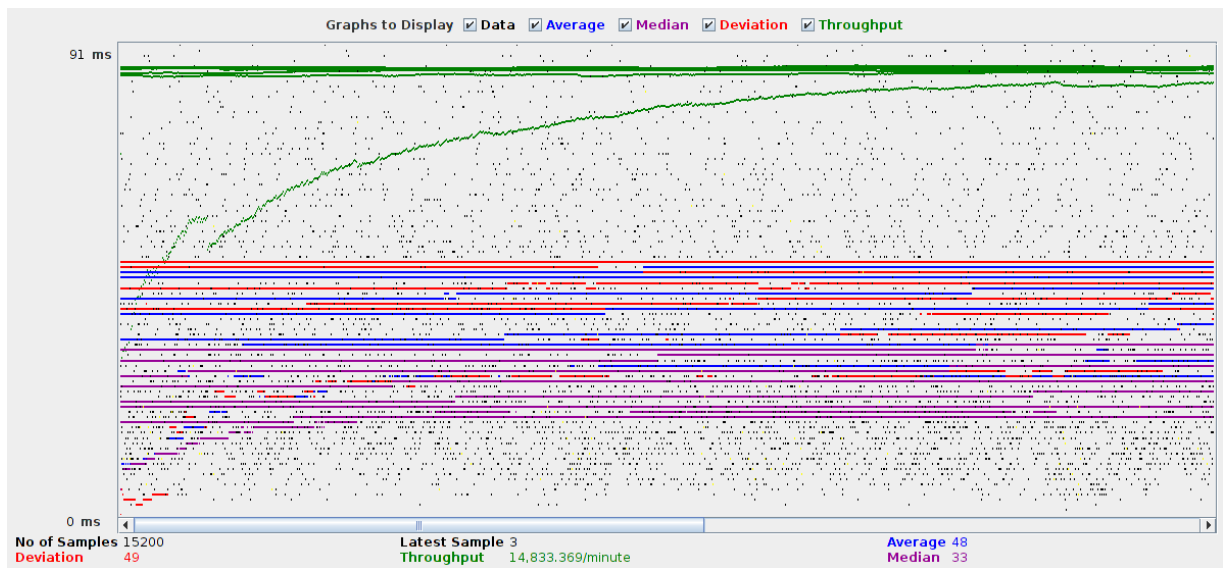
Server	Tên đại lượng tải	Giá trị tải
Mongos (Routing )	CPU	30 → 39%
	RAM	145/490 MB
Mongod Shard 0	CPU	25 → 40 %
	RAM	136/490
Mongod Shard 1	CPU	30 → 39%
	RAM	127/490 MB
Nginx server	CPU	45 → 60 %
	RAM	112/490MB
Tomcat Server 1	CPU	75 → 95 %
	RAM	357/994MB
Tomcat server 2	CPU	90 → 97 %
	RAM	266/994

*Bảng 11: Thông tin server sử dụng 2 tomcat server với 800 user*

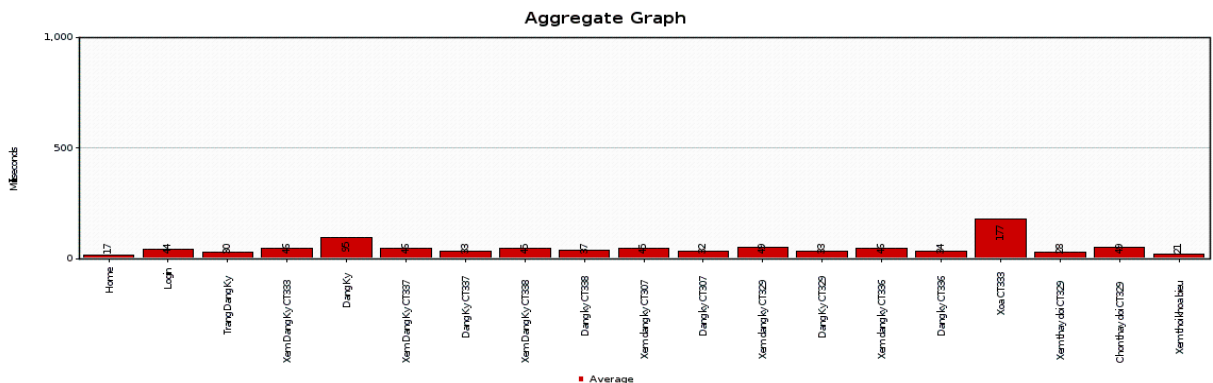
Sau đây là một số kết quả thu được từ công cụ Jmeter:

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	KB/sec
Home	800	17	12	37	52	89	2	158	0.00%	13.2/sec	43.5
Login	800	44	36	84	102	144	5	199	0.00%	13.2/sec	72.6
Trang Đăng Ký	800	30	25	59	70	100	4	206	0.00%	13.1/sec	132.3
Xem Đăng Ký ...	800	46	41	86	107	136	5	205	4.25%	13.1/sec	143.3
Dang Ky	800	95	50	229	268	328	2	451	0.00%	13.1/sec	2.8
Xem Dang Ky ...	800	46	39	89	106	136	5	189	3.50%	13.1/sec	147.5
Dang Ky CT337	800	33	29	63	76	116	2	179	0.00%	13.1/sec	2.9
Xem Dang Ky ...	800	45	38	82	96	143	5	181	3.12%	13.1/sec	147.9
Dang ky CT338	800	37	29	72	93	130	2	262	0.00%	13.1/sec	2.8
Xem dang ky ...	800	45	38	86	104	145	5	205	3.75%	13.1/sec	147.6
Dang ky CT307	800	32	26	64	83	114	1	182	0.00%	13.1/sec	2.8
Xem dang ky ...	800	49	43	89	113	145	6	174	3.38%	13.1/sec	159.6
Dang Ky CT329	800	33	28	67	79	120	2	182	0.00%	13.1/sec	2.9
Xem dang ky ...	800	46	38	84	103	148	6	179	3.50%	13.1/sec	147.4
Dang ky CT336	800	34	28	64	77	122	2	162	0.00%	13.1/sec	2.9
Xoa CT333	800	177	170	263	290	343	2	485	0.00%	13.1/sec	2.8
Xem thay doi ...	800	28	22	57	74	104	4	252	5.25%	13.1/sec	29.2
Chon thay do...	800	49	42	89	108	163	4	246	4.62%	13.1/sec	22.1
Xem thoi kho...	800	21	17	42	52	86	3	125	0.00%	13.1/sec	76.5
TOTAL	15200	48	33	102	156	251	1	485	1.65%	247.2/sec	1279.5

Bảng 12: Kết quả trả về của kịch bản 800 user cùng lúc sử dụng hai tomcat server



Biểu đồ 5: Kết quả 800 user cùng lúc sử dụng hai tomcat server



Biểu đồ 6: Thời gian xử lý request 800 user cùng lúc sử dụng hai Tomcat server



**Nhận xét:** Qua đồ thị, số liệu từ bảng của công cụ jmeter và thông tin hiệu suất từ server cho ta thấy khi 800 user cùng lúc thì việc xử lý request sẽ lâu hơn (xem đồ thị thời gian xử lý request). Và hai Tomcat server trong tình trạng quá tải (hiệu suất của CPU gần tới ngưỡng). Nhưng các server Database vẫn hoạt động tốt. Lượng request lỗi tăng (các lỗi đa phần là lỗi mất session) do lượng người dùng truy cập khá cao.

**Kết luận:** để hệ thống cluster có thể đáp ứng một lượng người dùng lớn 800 user thì phải mở rộng cluster bằng cách tăng cường Tomcat server vào hệ thống. Nâng cao đường truyền giữa các server trong cluster để hạn chế các request lỗi.

### **3.3.2. Nhận xét chung**

Thông qua những quá trình thực hiện với số lượng user và số lượng tomcat server khác nhau, chúng ta có thể thấy rằng số lượng tomcat server có ảnh hưởng lớn đến thời gian xử lý tất cả các request từ client. Qua đó ta thấy được số lượng tomcat server tăng lên thì số lượng user truy cập cùng lúc vào cluster cũng tăng lên theo.

Nhưng việc tăng các Tomcat server lên cũng phải đồng thời xem xét server CSDL hoạt động như thế nào? Nếu các server CSDL không đáp ứng nhanh các yêu cầu từ các Tomcat Server, thì chúng ta có thể mở rộng hệ thống cluster bằng cách tăng cường thêm server CSDL.



## **PHẦN: KẾT LUẬN**

### **I. KẾT QUẢ ĐẠT ĐƯỢC**

#### **1.1. Kết quả**

- Hiểu được hệ thống cluster và cơ chế hoạt động.
- Xây dựng hệ thống cluster: môi trường cài đặt, mô hình cluster.
- Thấy được các mối quan hệ tương quan giữa các giá trị tài nguyên với số lượng người dùng khi tham gia đăng ký học phần.
- Cấu hình Reverse Proxy với Nginx.
- Cấu hình cân bằng tải (load balancing) với Nginx.
- Giải quyết được vấn đề session khi load balancing bằng cách dùng Replication Session với Tomcat server.
- Hiểu được cơ chế hoạt động của MongoDB và các chức năng Index, Shard, Replication,... làm tăng tốc độ đọc, ghi file của MongoDB. Đảm bảo tính sẵn dùng cao của Database.

#### **1.2. Hạn chế**

- Chưa được áp dụng ngoài thực tiễn chỉ sử dụng công cụ mô phỏng (Jmeter).
- Chưa được áp dụng trên server thật chỉ sử dụng server ảo.
- Website đăng ký học phần chỉ mang tính chất “thời vụ” nên lãng phí tài nguyên.
- Để xây dựng hệ thống cluster cần một số lượng nhiều server tốn nhiều chi phí cho cluster.
- Hệ thống cluster chưa được cấu hình bảo mật cao.
- Việc Replication session của tomcat làm cho IO network tăng lên, do multicast các thông điệp trao đổi session giữa các servers, làm tăng traffic lên, ảnh hưởng đến bandwidth của mạng.
- Cần một hệ thống mạng với tốc độ đường truyền cao để tăng trao đổi dữ liệu giữa các server.

### **II. HƯỚNG PHÁT TRIỂN**

Từ những hạn chế chúng ta phát triển đề tài như sau:

- Đưa hệ thống cluster áp dụng vào thực tiễn nhằm để nâng cao tính thực dụng cho cluster.
- Xây dựng hệ thống cluster bằng server thật để tăng khả năng chịu tải, tính sẵn dùng cao hơn so với máy ảo.

- Thay vì sử dụng công cụ mô phỏng Jmeter chúng ta có thể đưa mô hình cluster hoạt động thực tiễn nhằm mang lại kết quả chính xác hơn.
- Muốn cho hệ thống cluster đáp ứng nhiều user cùng một lúc phải tăng cường số lượng server trong cluster.
- Tăng độ đường truyền giữa server trong cluster nhằm nâng cao khả năng trao đổi dữ liệu giữa các server khác nhau. Hạn chế được việc mất session do đường truyền không tốt giảm thiểu request lỗi.
- Cấu hình bảo mật cho hệ thống cluster.
- Cần có đội ngũ quản trị cluster chuyên nghiệp, để có thể xử lý nhanh các xử lý trong cluster.

## TÀI LIỆU THAM KHẢO

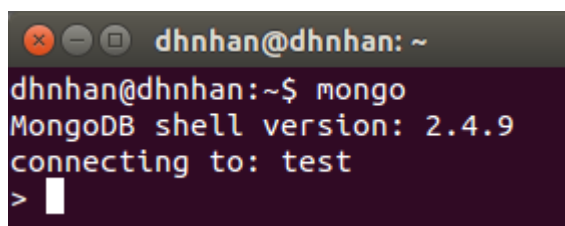
- [1] <https://technet.microsoft.com/en-us/library/cc785197%28v=ws.10%29.aspx>
- [2] <http://wiki.nginx.org/Main>
- [3] <http://nginx.com/resources/admin-guide/reverse-proxy/>
- [4] <http://nginx.com/resources/admin-guide/load-balancer/>
- [5] <https://tomcat.apache.org/tomcat-7.0-doc/cluster-howto.html>
- [6] <http://www.ramkitech.com/2012/11/tomcat-clustering-series-part-3-session.html>
- [7] <http://docs.mongodb.org/manual/tutorial/create-an-index/>
- [8] <http://docs.mongodb.org/manual/sharding/>
- [9] <https://www.digitalocean.com/community/tutorials/how-to-create-a-sharded-cluster-in-mongodb-using-an-ubuntu-12-04-vps>
- [10] <https://www.digitalocean.com/community/tutorials/how-to-implement-replication-sets-in-mongodb-on-an-ubuntu-vps>
- [11] <http://docs.mongodb.org/manual/replication/>
- [12] <http://jmeter.apache.org/>
- [13] [http://www.testervn.com/#page\\_1/](http://www.testervn.com/#page_1/)
- [14] <http://code.freetuts.net/mongodb>
- [15] Tham khảo luận văn “Đánh giá tải Moodle” của Quách Kim Hải

## PHỤ LỤC

### PHỤ LỤC A: MONGODB

#### 1. Cài đặt MongoDB trên Ubuntu Server 14.04

- Mở terminal (Ctrl + T)
- `sudo apt-get update`
- `sudo apt-get dist-upgrade`
- `sudo apt-get install -y mongodb`
- Sao đó kiểm tra thành công hay chưa bằng cách gõ mongo:



```
dhnhan@dhnhan: ~  
dhnhan@dhnhan:~$ mongo  
MongoDB shell version: 2.4.9  
connecting to: test  
>
```

#### 2. Index

Chỉ mục làm tăng hiệu suất truy vấn lên rất nhiều. Điều quan trọng là nghĩ xem xét tất cả các loại truy vấn cần trong ứng dụng để xác định những chỉ mục liên quan. Khi đã xác định xong, việc tạo ra các chỉ mục trong MongoDB là khá dễ dàng.

##### a. Các khái niệm cơ bản

Chỉ mục là một cấu trúc dữ liệu, thu thập thông tin về giá trị của các trường trong các văn bản của một bộ sưu tập. Cấu trúc dữ liệu này được sử dụng trong tối ưu truy vấn Mongo để sắp xếp nhanh các văn bản trong một bộ sưu tập.

##### b. Đánh chỉ mục

Chúng ta có thể khởi tạo chỉ mục bằng cách gọi hàm `ensureIndex()` và cung cấp một văn bản với một hoặc nhiều khóa để đánh chỉ mục. Ví dụ đánh chỉ mục cho trường name trong students

```
db.student.ensureIndex({mssv:1});
```

Hàm `ensureIndex()` chỉ khởi tạo chỉ mục nếu nó chưa tồn tại. Để kiểm tra việc tồn tại chỉ mục trên bộ sưu tập students, ta có thể chạy hàm `db.student.getIndexes()`.

Khi một bộ sưu tập được đánh chỉ mục trên một khóa nào đó, truy cập ngẫu nhiên trên biểu thức truy vấn có chứa khóa đó sẽ được thực hiện rất nhanh. Nếu không được đánh chỉ mục, MongoDB phải soát tất cả các văn bản để kiểm tra giá trị của khóa đó trong truy vấn.

Chỉ mục mặc định: Một chỉ mục luôn luôn được tạo ra là `_id`. Chỉ mục này là đặc biệt và không thể bị xóa. Chỉ mục `_id` là duy nhất cho các khóa của nó.

Đánh chỉ mục trong khóa nhúng: Với MongoDB chúng ta thậm chí có thể đánh chỉ mục trên các khóa bên trong văn bản nhúng. Ví dụ

```
db.student.ensureIndex({"address.city": 1})
```

#### c. Xóa chỉ mục

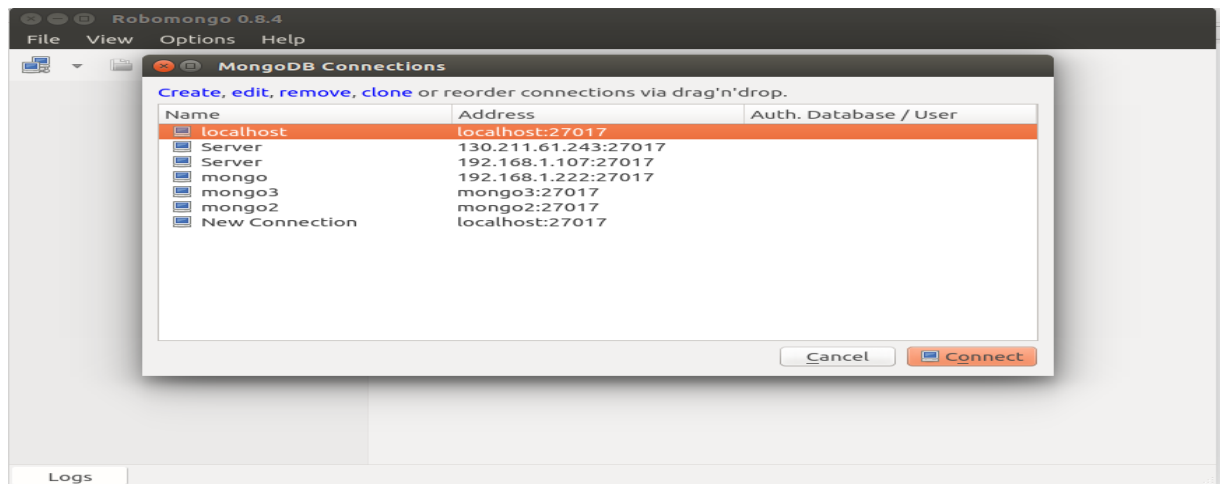
- Xóa tất cả các chỉ mục trên bộ sưu tập: `db.collection.dropIndexes();`
- Xóa chỉ mục đơn: `db.collection.dropIndex({x: 1, y: -1})`
- Chạy trực tiếp như một lệnh mà không cần hỗ trợ:  
// note: command was "deleteIndexes", not "dropIndexes", before MongoDB v1.3.2  
// remove index with key pattern {y:1} from collection foo  
`db.runCommand({dropIndexes:'foo', index : {y:1}})`  
// remove all indexes:  
`db.runCommand({dropIndexes:'foo', index : '*'})`

#### d. Hiệu suất đánh chỉ mục

Việc đánh chỉ mục thực hiện rất nhanh. Cập nhật được thực hiện nhanh hơn vì MongoDB có thể tìm thấy các văn bản cần cập nhật rất nhanh chóng. Tuy nhiên, với việc sử dụng chỉ mục, khi ghi dữ liệu vào bộ sưu tập, các khóa sau đó phải được thêm vào trường chỉ mục. Như vậy, chỉ mục chỉ tốt cho bộ sưu tập có số lượng đọc nhiều hơn rất nhiều số lượng ghi. Đối với các bộ sưu tập chú trọng ghi, việc sử dụng chỉ mục, trong một số trường hợp, có thể phản tác dụng. Hầu hết các bộ sưu tập đều chú trọng đọc, vì vậy mà chỉ mục là tốt trong hầu hết các tình huống.

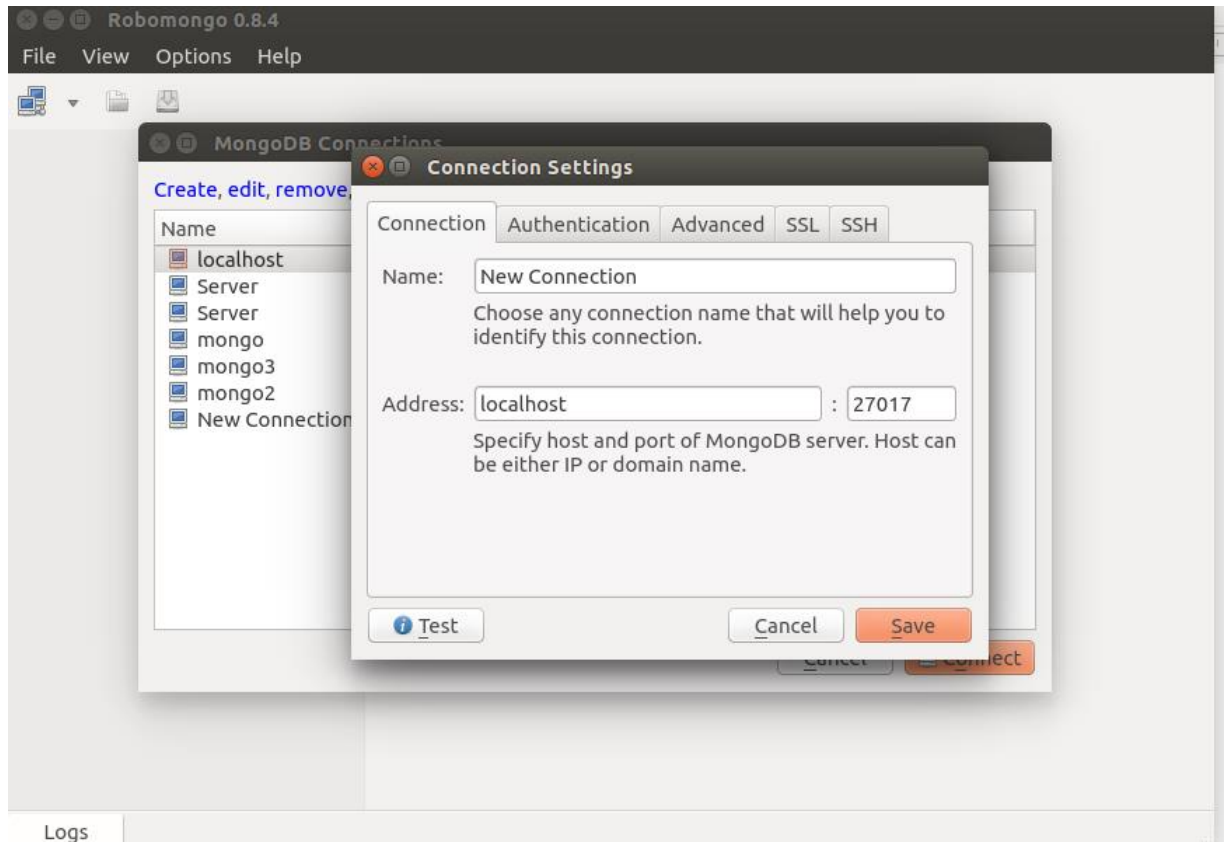
### 3. Truy vấn

Giống như CSDL quan hệ, MongoDB cũng hỗ trợ truy vấn với các câu điều kiện phức tạp. Robomongo là một công cụ cho phép thiết lập kết nối và thực hiện các truy vấn cũng như hiển thị kết quả với các câu truy vấn MongoDB, hình dưới là giao diện tổng quát của Robomongo.



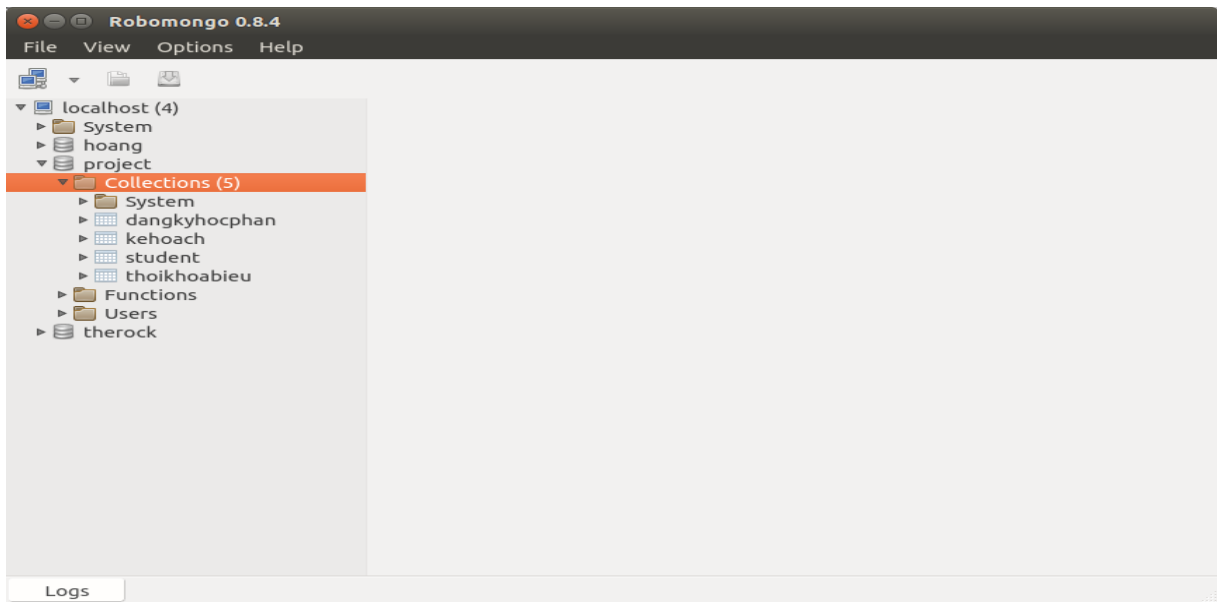
Giao diện tổng quát của Robomongo

Chọn File → connect → create. Điền thông tin name, địa chỉ mongo server và port

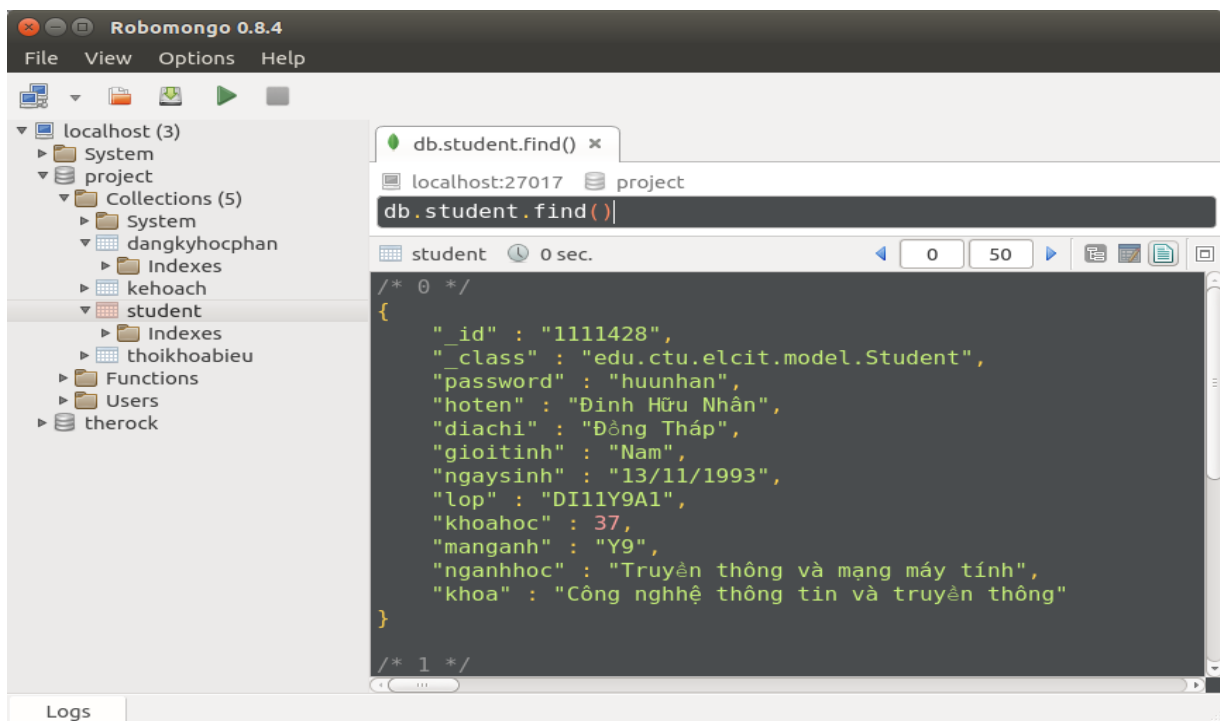


Tạo mới connetion

Sau khi tạo thành công ta được:



Để hiển thị toàn bộ dữ liệu của một document (table) của một database, chọn vào document và chọn “View Documents”.



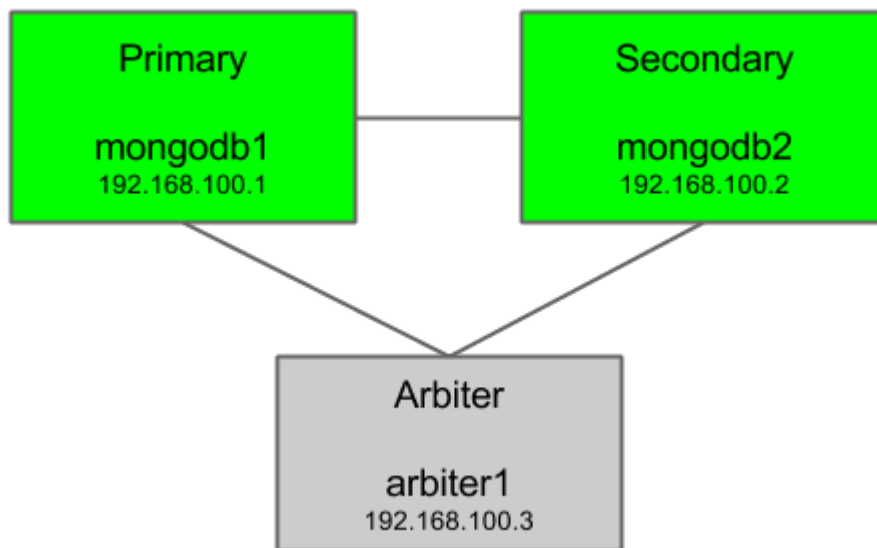
Sau đây là phần so sánh giữa các câu lệnh truy vấn cơ bản của CSDL quan hệ (mysql) và MongoDB.



SQL Statement	Mongo Statement
CREATE TABLE USERS (a Number, b Number)	implicit; can also be done explicitly with db.createCollection("mycoll")
ALTER TABLE users ADD ...	implicit
INSERT INTO USERS VALUES(3,5)	db.users.insert({a:3,b:5})
SELECT a,b FROM users	db.users.find({}, {a:1,b:1})
SELECT * FROM users	db.users.find()
SELECT * FROM users WHERE age=33	db.users.find({age:33})
SELECT a,b FROM users WHERE age=33	db.users.find({age:33}, {a:1,b:1})
SELECT * FROM users WHERE age=33 ORDER BY name	db.users.find({age:33}).sort({name:1})
SELECT * FROM users WHERE age>33	db.users.find({age:{\$gt:33}})
SELECT * FROM users WHERE age!=33	db.users.find({age:{\$ne:33}})
SELECT * FROM users WHERE name LIKE "%Joe%"	db.users.find({name:/Joe/})
SELECT * FROM users WHERE name LIKE "Joe%"	db.users.find({name:/^Joe/})
SELECT * FROM users WHERE age>33 AND age<=40	db.users.find({'age':{\$gt:33,\$lte:40}})
SELECT * FROM users ORDER BY name DESC	db.users.find().sort({name:-1})
SELECT * FROM users WHERE a=1 and b='q'	db.users.find({a:1,b:'q'})
SELECT * FROM users LIMIT 10 SKIP 20	db.users.find().limit(10).skip(20)

<pre>SELECT * FROM users WHERE a=1 or b=2</pre>	<pre>db.users.find( { \$or : [ { a : 1 }, { b : 2 } ] } )</pre>
<pre>SELECT * FROM users LIMIT 1</pre>	<pre>db.users.findOne()</pre>
<pre>SELECT DISTINCT last_name FROM users</pre>	<pre>db.users.distinct('last_name')</pre>
<pre>SELECT COUNT(*y) FROM users</pre>	<pre>db.users.count()</pre>
<pre>SELECT COUNT(*y) FROM users where AGE &gt; 30</pre>	<pre>db.users.find({age: {'\$gt': 30}}).count()</pre>
<pre>SELECT COUNT(AGE) from users</pre>	<pre>db.users.find({age: {'\$exists': true}}).count()</pre>

#### 4. Hướng dẫn replication mongodb



- sửa file /etc/hosts trên 3 server mongodb
  - o 192.168.100.1 mongo1
  - o 192.168.100.2 mongo2
  - o 192.168.100.3 mongo3
- Tắt dịch vụ monodb : `sudo service mongo stop`

- Cả 3 server đều tạo thư mục /data/db/ : `sudo mkdir -R /data/db/`
  - Cấu hình trên cả 3 server mongo
    - o thay đổi file /etc/mongo.conf
      - `dbpath=/data/db/`
      - `port = 27017`
      - `replSet = rs0`
      - `fork = true`
    - o config lại file mongo.conf: `mongod -config /etc/mongo.conf`
  - Vào server 1:
    - o Gõ: `mongo`
    - o `rs.initiate()`
    - o Xem thông tin của replication: `rs.config()`

```
{
  "_id" : "rs0"
  "version" : 1,
  "members" : [
    {
      "_id" : 0,
      "host" : "mongo0:27017"
    }
  ]
}
```
  - Sau đó thêm các member vào:
    - o `rs.add(mongo2:27017)`
    - o `rs.add(mongo2:27017)`
  - Xem lại `rs.config()`
- ```
{
  "_id" : "rs0"
  "version" : 1,
  "members" : [
    {
      "_id" : 0,
      "host" : "mongo0:27017"
    },
    {
      "_id" : 1,
      "host" : "mongo1:27017"
    }
  ]
}
```

```
    }  
    {  
      "_id" : 2,  
      "host" "mongo2:27017"  
    }  
  ]  
}
```

## 5. Cấu hình sharding với mongod

Chuẩn bị các server có cấu hình như sau:

| Service                    | Daemon | IP            | Port  | dbpath        |
|----------------------------|--------|---------------|-------|---------------|
| Shard controller (routing) | mongos | 192.168.1.222 | 27017 | /data/db/     |
| Config server              | mongod | 192.168.1.222 | 27018 | /data/config/ |
| Shard 0                    | mongod | 192.168.1.223 | 27017 | /data/db/     |
| Shard 1                    | mongod | 192.168.1.224 | 27017 | /data/db/     |

- Thêm vào file hosts trên 3 server: *sudo nano /etc/hosts*
  - 192.168.1.222 mongo1
  - 192.168.1.223 mongo2
  - 192.168.1.224 mongo3
- Khi chạy không được stop tiến trình .
- Cấu hình config server: *mongod --port 27018 --dbpath /data/config/*
- Cấu hình mongos: *mongos --configdb mongo1:27018 --port 27017 --chunkSize 1*
- Cấu hình shard 0: *mongod --port 27017 --dbpath /data/db/ --shardsvr*
- Cấu hình shard 1: *mongod --port 27017 --dbpath /data/db/ --shardsvr*

## PHỤ LỤC B: NGINX SERVER

### 1. Cài đặt Nginx

- Mở terminal (Ctrl + T)
- `sudo apt-get update`
- `sudo apt-get install nginx`
- Kiểm tra trạng thái hoạt động của nginx: `sudo service nginx status`

### 2. Cấu hình cơ bản Nginx

Các module cơ bản cung cấp các khai báo cho phép chúng ta định nghĩa các tham số của các chức năng cơ bản của Nginx. Có 3 module cơ bản là:

- **Core module:** cung cấp các chức năng như quản lý tiến trình và bảo mật.
- **Events module:** cho phép chúng ta cấu hình các cơ chế bên trong của khả năng kết nối mạng trong Nginx.
- **Configuration module:** cho phép cơ chế đính kèm các tập tin cấu hình khác vào bên trong tập tin cấu hình chính của Nginx (thể hiện qua khai báo *include*).

### 3. Các thông số của Nginx

#### a. Worker\_processes

Với cấu hình mặc định, Nginx sẽ sử dụng một CPU để xử lý các tác vụ của mình. Tùy theo mức độ hoạt động của web server mà chúng ta có thể thay đổi lại thiết lập này. Ví dụ với các web server hay sử dụng về SSL, gzip thì ta nên đặt chỉ số của worker\_processes này lên cao hơn. Nếu website của bạn có số lượng các tệp tin tĩnh nhiều, và dung lượng của chúng lớn hơn bộ nhớ RAM thì việc tăng worker\_processes sẽ tối ưu bằng thông đĩa của hệ thống.

Để xác định số cores của CPU của hệ thống ta có thể thực hiện lệnh:

```
# cat /proc/cpuinfo | grep processor
```

```
processor : 0
```

```
processor : 1
```

```
processor : 2
```

```
processor : 3
```

Như ở trên, CPU của chúng ta có 4 cores. Để thay đổi mức sử dụng CPU của nginx ta sửa tệp tin cấu hình chính.

```
vi /etc/nginx/nginx.conf
```

```
worker_processes 4;
```

*b. Worker\_connections*

Worker\_connections sẽ cho biết số lượng connection mà CPU sẽ xử lý. Mặc định, số lượng connection này được thiết lập là 1024. Để xem về mức giới hạn sử dụng của hệ thống bạn có thể dùng lệnh ulimit. Con số thiết lập của worker\_connections nên nhỏ hơn hoặc bằng giới hạn này! Nếu bạn đã điều chỉnh lại giá trị worker\_processes giúp Nginx sử dụng nhiều cores để xử lý các tác vụ hơn thì có thể thêm dòng cấu hình sau để tăng số lượng clients lên cao nhất.

`max_clients = worker_processes * worker_connections`

*c. Buffer*

Một trong những cấu hình quan trọng để tối ưu Nginx là thiết đặt các giá trị buffer. Nếu bạn thiết lập bộ nhớ buffer quá nhỏ thì sẽ dễ dẫn tới tình trạng “thất cổ chai” khi web server của chúng ta tiếp nhận một lượng traffic lớn. Để thay đổi các giá trị buffer này, chúng ta có thể thêm vào các dòng cấu hình ở thẻ http của file cấu hình chính nginx.conf.

`client_body_buffer_size 8K;`

`client_header_buffer_size 1k;`

`client_max_body_size 2m;`

`large_client_header_buffers 2 1k;`

Trong đó:

- **client\_body\_buffer\_size:** Thiết đặt giá trị kích thước của body mà client yêu cầu. Nếu kích thước được yêu cầu lớn hơn giá trị buffer thì sẽ được lưu vào temporary file.
- **client\_header\_buffer\_size:** Thiết đặt giá trị kích thước của header mà client yêu cầu. Thông thường thì kích thước này 1K là đủ.
- **client\_max\_body\_size:** Thiết đặt giá trị kích thước tối đa của body mà client có thể yêu cầu được, xác định bởi dòng Content-Length trong header. Nếu kích thước body yêu cầu vượt giới hạn này thì client sẽ nhận được thông báo lỗi “Request Entity Too Large” (413).
- **large\_client\_header\_buffers:** Thiết đặt giá trị kích về số lượng và kích thước lớn nhất của buffer dùng để đọc các headers có kích thước lớn từ các request của client. Nếu client gửi một header quá lớn Nginx sẽ trả về lỗi “Request URL too large” (414) hoặc “Bad request” (400) nếu header của request quá dài.

Ngoài ra chúng ta cũng cần thiết đặt lại các giá trị timeout để tối ưu hiệu suất hoạt động của web server với các client:

```
client_body_timeout 10;  
client_header_timeout 10;  
keepalive_timeout 15;  
send_timeout 10;
```

Trong đó:

- **client\_body\_timeout:** Thiết đặt thời gian tải body của webpage từ client. Nếu quá thời gian này, client sẽ nhận thông báo trả về “Request time out” (408).
- **client\_header\_timeout:** Thiết đặt thời gian tải title của webpage từ client. Nếu quá thời gian này, client sẽ nhận thông báo trả về “Request time out” (408).
- **keepalive\_timeout:** Thiết đặt thời gian sống của kết nối từ client, nếu quá thời gian này thì kết nối sẽ bị đóng.
- **send\_timeout:** Thiết đặt thời gian phản hồi dữ liệu giữa client và server, nếu quá thời gian này thì nginx sẽ tắt kết nối.

d. Nén các gói dữ liệu gửi đi bằng Gzip

Gzip sẽ giúp nén các dữ liệu trước khi chuyển chúng tới Client. Đây là một cách để tăng tốc độ truy cập website của chúng ta. Trong thẻ http của file cấu hình chính nginx.conf ta có thể thêm.

```
gzip on;  
gzip_comp_level 2;  
gzip_min_length 1000;  
gzip_proxied expired no-cache no-store private auth;  
gzip_types text/plain application/xml;  
gzip_disable "MSIE [1-6]\.";
```

e. Cache nội dung các tệp tĩnh

Hầu hết các request từ client tới website của chúng ta để load các nội dung như: hình ảnh, java script, css, flash,... Chúng ta nên thực hiện việc lưu cache lại các tệp tin có nội dung tĩnh này trên Nginx.

```
location ~* "\.(js|ico|gif|jpg|png|css|html|htm|swf|htc|xml|bmp|cur)$" {  
    root /home/site/public_html;  
    add_header Pragma "public";  
    add_header Cache-Control "public";  
    expires 3M;  
    access_log off;  
    log_not_found off;}
```



*f. Các thông số cơ bản của caching*

```
http {
    proxy_cache_path /var/www/cache levels=2 keys_zone=my-cache:8m
    max_size=1000m inactive=600m;
    proxy_temp_path /var/www/cache/tmp;
    server {
        location / {
            proxy_pass http://example.net;
            proxy_cache my-cache;
            proxy_cache_valid 200 302 60m;
            proxy_cache_valid 404 1m;
        }
    }
}
```

Trong đó:

- **/var/www/cache** là thư mục chứa cache.
- **Level =2** phân cấp thư mục lưu cache.
- **key\_zone=cache\_one:8m** tên của cache zone là cache\_one.
- **Inactive =1d**: thành phần cache nào không được truy cập trong vòng 1 ngày sẽ bị xóa.
- **proxy\_cache cache\_one** : chỉ định cache vào zone cache\_one
- **proxy\_cache\_valid 200 15m**: các request có file tồn tại (200) sẽ có giá trị cache trong vòng 15m.
- **proxy\_cache\_key \$host\$request\_uri** : chỉ định key cache, giúp nginx lần được cache của một URL ở đâu trong thư mục cache.
- **proxy\_set\_header X-Forwarded-For \$remote\_addr**: dùng để thêm một dòng trong header trả về cho client.
- **proxy\_ignore\_headers** : thông báo cho nginx bỏ qua những header nào của client. Trong các header của người dùng có thể có những trường để điều khiển cache (cache control) nên chúng ta cần báo cho nginx biết bỏ qua những thông số này để việc cache được kiểm soát tối đa.
- **proxy\_read\_timeout Default: 60s**

#### 4. Cấu hình liên quan tới Http

Module HTTP Core là thành phần chứa tất cả các khối, chỉ thị và các biến cơ bản của máy chủ HTTP. Mặc định module này được cài đặt trong khi biên dịch, nhưng không được bật lên khi Nginx chạy, việc sử dụng module này là không bắt buộc. Chúng ta có thể gỡ bỏ nó trong quá trình biên dịch chương trình.



Module này là 1 trong những module tiêu chuẩn lớn nhất của Nginx – nó cung cấp 1 số lượng ấn tượng các chỉ thị và biến. Để có thể hiểu được tất cả các yếu tố mới này và vai trò của chúng, chúng ta cần hiểu tổ chức luận lý được giới thiệu trong 3 khối chỉ thị chính – http, server và location.

## HTTP , SERVER VÀ LOCATION

Đây là các directive quản lý module http core:

- **http**: được khai báo ở phần đầu của tập tin cấu hình. Nó cho phép chúng ta bắt đầu định nghĩa các chỉ thị và các khối từ tất cả các module liên quan đến khía cạnh HTTP của Nginx. Khối chỉ thị này có thể được khai báo nhiều lần trong tập tin cấu hình, và các giá trị chỉ thị được chèn trong khối http sau sẽ ghi đè lên các chỉ thị nằm trong khối http trước đó.
- **server**: khối này cho phép chúng ta khai báo 1 website. Nói cách khác, 1 website cụ thể (được nhận diện bởi 1 hoặc nhiều hostname) được thừa nhận bởi Nginx và nhận cấu hình của chính nó. Khối này chỉ có thể được dùng bên trong khối http.
- **location**: cho phép chúng ta định nghĩa 1 nhóm các thiết lập được áp dụng cho 1 vị trí cụ thể trên website (thể hiện qua URL của website đó). Khối location có thể được dùng bên trong 1 khối server hoặc nằm chồng bên trong 1 khối location khác.

### a. Cấu hình virtual hosting

Như ở phần trên ta đã cấu hình server Nginx phục vụ host name `www.dhnhan.com`. Và Nginx hỗ trợ tính năng virtual hosting , bây giờ ta có nhu cầu phục vụ thêm host name `www.dhnhan.net`. Ta sẽ cấu hình như sau.

Ở đây ta sử dụng thêm một directive server được đặt trong khối http:

```
server {  
    listen 80;  
    server_name www.dhnhan.net dhnhan.net;  
    location / {  
        root html;  
        index second.html;  
    }  
    location /image/ {  
        root images;  
    }  
}
```

}

Nginx sẽ quyết định xử lý request theo host name nào phụ thuộc vào trường host name trong gói tin http, giả sử ta có url: `www.dhnhan.net` khi gói tin được gửi đến server Nginx dựa vào host name là `www.dhnhan.net` để quyết định gói request sẽ được đáp ứng ở đâu. (Cấu hình virtual host theo kỹ thuật `Name_based`).

## 5. Cấu hình Reverse Proxy và Load balancing

Chuẩn bị các server:

- Nginx server: IP: 192.168.1.202, Port: 80
- Tomcat server 1: IP: 192.168.1.199, port 8181
- Tomcat server 2: IP: 192.168.1.201, port 8282

Sau khi cài đặt Nginx thành công

Tạo một cluster trong nginx thêm vào trong khối http: `sudo nano /etc/nginx/nginx.conf`

```
upstream loadbalancer{
    #ip_hash;
    #sticky;
    server 192.168.1.199:8181;
    server 192.168.1.201:8282;
}
```

Tạo một virtual server trong khối http

```
server {
    listen 80;
    server_name 192.168.1.202;
    location ~*^.+.(jpg|jpeg|gif|png|ico|css|zip|tgz|gz|rar|bz2|doc|xls|exe|pdf|ppt|txt|tar|mid|midi|wav|bmp|rtf|js)$ {
        root /var/static;
        expires -1;
        access_log off;
        add_header Cache-Control "public";
    }
    # Main location
    location / {
        proxy_pass http://loadbalancer/;
        proxy_redirect off;
    }
}
```

```
proxy_set_header Host          $host;
proxy_set_header X-Real-IP      $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
client_max_body_size 10m;
client_body_buffer_size 128k;
proxy_connect_timeout 90;
proxy_send_timeout 90;
proxy_read_timeout 90;
proxy_buffer_size 4k;
proxy_buffers 4 32k;
proxy_busy_buffers_size 64k;
proxy_temp_file_write_size 64k;
}
}
```

Giải thích các thành phần cơ bản:

- listen 80; # lắng nghe ở cổng 80
- server\_name 192.168.1.202; # khai báo địa chỉ IP hoặc tên miền
- location ~\*  
^.+.(jpg|jpeg|gif|png|ico|css|zip|tgz|gz|rar|bz2|doc|xls|exe|pdf|ppt|txt|tar|mid|midi|wav|bmp|rtf|js)\$ # định nghĩa proxy, những file có phần mở trong đây sẽ được cache lại nginx server và được lưu trong /var/static
- proxy\_pass <http://loadbalancer/>; # chuyển request đến cluster

## 6. Cấu hình replication session Tomcat Server 7

Tomcat server 1 : 192.168.1.199:8181

Tomcat server 2 : 192.168.1.201:8282

```
– Thêm một cluster vào file server.xml trong thư mục Tomcat/conf/server.xml
<Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"
channelSendOptions="8">
    <Manager className="org.apache.catalina.ha.session.DeltaManager"
        expireSessionsOnShutdown="false"
        notifyListenersOnReplication="true"/>

    <Channel className="org.apache.catalina.tribes.group.GroupChannel">
        <Membership
className="org.apache.catalina.tribes.membership.McastService"
        address="228.0.0.4"
        port="45564"
```

```
frequency="500"
dropTime="3000"/>

    <Sender
className="org.apache.catalina.tribes.transport.ReplicationTransmitter">
    <Transport
className="org.apache.catalina.tribes.transport.nio.PooledParallelSender"/>
    </Sender>

    <Receiver
className="org.apache.catalina.tribes.transport.nio.NioReceiver"
address="auto"
port="4000"
autoBind="100"
selectorTimeout="5000"
maxThreads="6"/>

    <Interceptor
className="org.apache.catalina.tribes.group.interceptors.TcpFailureDetector"/>

    <Interceptor
className="org.apache.catalina.tribes.group.interceptors.MessageDispatch15Interce
ptor"/>

</Channel>

<Valve className="org.apache.catalina.ha.tcp.ReplicationValve" filter=""/>

<Valve className="org.apache.catalina.ha.session.JvmRouteBinderValve"/>

    <ClusterListener
className="org.apache.catalina.ha.session.JvmRouteSessionIDBinderListener"/>

    <ClusterListener
className="org.apache.catalina.ha.session.ClusterSessionListener"/>

</Cluster>
```

Tương tự với Tomcat server 2: nhưng phải thay đổi port

```
<Receiver className="org.apache.catalina.tribes.transport.nio.NioReceiver"  
    address="auto"  
    port="4001"  
    autoBind="100"  
    selectorTimeout="5000"  
    maxThreads="6"/>
```

Sau đó thay đổi file host của Tomcat server 1: *sudo nano /etc/hosts*

- 192.168.1.199 localhost

Thay đổi file host của Tomcat server 2: *sudo nano /etc/hosts*

- 192.168.1.201 localhost

Thêm thẻ *<distributable/>* vào source web ở file web.xml

## PHỤ LỤC C: CÀI ĐẶT VÀ THIẾT LẬP MÔI TRƯỜNG JAVA

### 1. Cài đặt JDK trên Ubuntu server 14.04

- Mở terminal: (Ctrl + T)
- *sudo apt-get update*
- Cài đặt JRE: *sudo apt-get install openjdk-7-jre*
- Cài đặt JDK: *sudo apt-get install openjdk-7-jdk*
- Kiểm tra xem cài đặt thành công : *java -version*

### 2. Cài đặt Tomcat server 7

- Download file: <https://tomcat.apache.org/download-70.cgi>
- Start dịch vụ Tomcat vào Terminal : *sudo Tomcat/bin/startup.sh*
- Stop dịch vụ Tomcat vào Terminal: *sudo Tomcat/bin/shutdown.sh*

### 3. Cài đặt Apache Jmeter

- Download file: [http://jmeter.apache.org/download\\_jmeter.cgi](http://jmeter.apache.org/download_jmeter.cgi)
- Giải nén file ra.

Name	Date modified	Type	Size
bin	4/26/2015 6:46 PM	File folder	
docs	3/8/2015 6:51 PM	File folder	
extras	3/8/2015 10:13 AM	File folder	
lib	3/8/2015 6:58 PM	File folder	
licenses	3/8/2015 7:02 PM	File folder	
printable_docs	3/8/2015 6:58 PM	File folder	
LICENSE	3/8/2015 7:02 PM	unisAlieS	11 KB
NOTICE	3/8/2015 7:02 PM	unisAlieS	1 KB
README	3/8/2015 7:02 PM	unisAlieS	7 KB

- Khởi động JMeter bằng cách chạy tập tin *jmeter.bat* trên hệ điều hành Windows hoặc *jmeter.sh* trên các hệ điều hành Linux. Giao diện sau khi cài đặt hoàn thành:

