TRƯỜNG ĐẠI HỌC CẦN THƠ

KHOA CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÁO CÁO THỰC TẬP THỰC TẾ
NGÀNH CÔNG NGHỆ THÔNG TIN
(CT450)

Giáo viên hướng dẫn:
Nguyễn Thanh Hải

Cán bộ hướng dẫn:
Lưu Trùng Dương
Lê Thanh Sang

Sinh viên thực hiện
Lê Quang Sang
B1606927

Cần Thơ, 07/2020

LỜI CẢM ƠN

Qua 2 tháng thực tập tại trung tâm em nhận thấy mình đã được học hỏi rất nhiều, giúp em bổ sung thêm các kiến thức còn thiếu sót. Em rất cảm ơn các thầy ở trung tâm đã giúp đỡ hỗ trợ em trong khoảng thời gian thực tập thực tế, cung cấp tài nguyên môi trường để em có thể tìm hiểu thêm nhiều công nghệ, kiến trúc phần mềm hiện tại.

Do kiến thức còn hạn hẹp nên không tránh khỏi những thiếu sót trong cách hiểu, lỗi trình bày. Em rất mong nhận được sự đóng góp ý kiến của thầy, cô trong trung tâm giúp em hoàn thiện bản thân nhiều hơn.

Em xin chân thành cám ơn!

PHIẾU ĐÁNH GIÁ BÁO CÁO KẾT QUẢ THỰC TẬP
HỌC KỲ 3 – 2019-2020
(Dùng cho giáo viên chấm báo cáo thực tập)

Họ và tên cán bộ chấm báo cáo: Nguyễn Thanh Hải

Họ tên sinh viên thực tập: Lê quang Sang Mã số SV: B1606927

Nội dung đánh giá	Điểm tối đa	Điểm chấm
I. Hình thức trình bày	**1.0**	
I.1 Đúng format của khoa (Trang bìa, trang lời cảm ơn, trang đánh giá thực tập của khoa, trang mục lục và các nội dung báo cáo). Sử dụng đúng mã và font tiếng Việt (Unicode Times New Roman, Size 13)	0.5	
I.2 Trình bày mạch lạc, súc tích, không có lỗi chính tả	0.5	
II. Phiếu theo dõi	**4.75**	
II.1 Có lịch làm việc đầy đủ cho 8 tuần	0.25	
II.2 Số buổi thực tập tại cơ quan trong 1 tuần >=6; ít hơn 6 buổi 0.0 điểm	1.0	
II.3 Hoàn thành tốt kế hoạch công tác ghi trong lịch làm việc. Cách tính điểm = (Điểm cộng của cán bộ hướng dẫn/100) x 3.5	3.5	
III. Nội dung thực tập (quyển báo cáo)	**4.25**	
III.1 Có được sự hiểu biết tốt về cơ quan nơi thực tập	0.5	
III.2 Phương pháp thực hiện phù hợp với nội dung công việc được giao	1.0	
III.3 Kết quả củng cố lý thuyết	0.5	
III.4 Kết quả rèn luyện kỹ năng thực hành	0.5	
III.5 Kinh nghiệm thực tiễn thu nhận được	0.5	
III.6 Kết quả công việc có đóng góp cho cơ quan nơi thực tập	1.25	
TỔNG CỘNG	**10.0**	
Điểm trừ		
Điểm còn lại		

Lưu ý:
Không dự họp để nghe phổ biến TTTT: trừ 1 điểm
Không gởi phiếu giao việc về khoa đúng hạn (đến 16/07/2020 theo dấu bưu điện): trừ 1 điểm

Ngày tháng năm
GV chấm báo cáo

MỤC LỤC

I. GIỚI THIỆU TỔ CHỨC TRUNG TÂM

Cơ cấu tổ chức trung tâm bao gồm một giám đốc điều hành toàn bộ trung tâm, một phó giám đốc chiệu trách nhiệm quản lý các tổ trong trung tâm, gồm các tổ: tổ văn phòng, tổ tông tin, tổ mạng máy tính, tổ hệ thống tích hợp.

Trung tâm quản trị mạng trường Đại học Cần Thơ với nhiệm vụ vận hành, quản lý, bảo trì xây dựng hệ thống hạ tầng mạng máy tính hệ thống websites của trường và các hệ thống trực tuyến của trường, quản lý hộ thống email, tài khoản mật khẩu của sinh viên, cán bộ trong trường.

II. NỘI DUNG CÔNG VIỆC

1. Tìm hiểu kiến trúc Oracle RAC
2. Triển khai kiến trúc Oracle RAC trên nền tảng Docker
3. Triển khai kiến trúc Oracle RAC trên nền tảng máy ảo Vmware
4. Triển khai cài đặt máy chủ upload tập tin trên nền tảng Docker Swarm

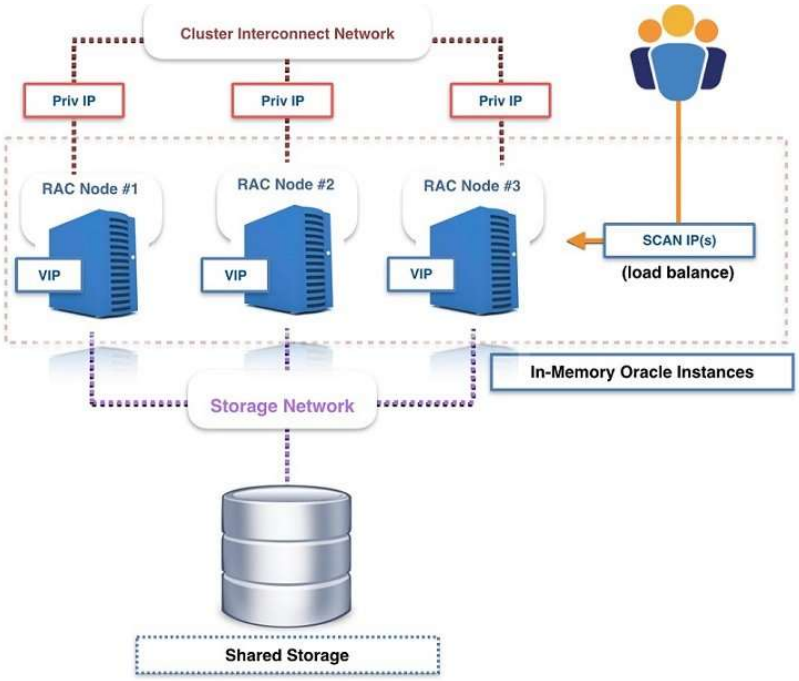
III. QUÁ TRÌNH THỰC HIỆN

Tiềm hiểu kiến trúc Oracle RAC

Truyền thống thì mỗi hệ quản trị cơ sỡ dữ liệu bao gồm một thể hiện cơ sở dữ liệu và một cơ sở dữ liệu bên dưới để lưu trữ tập tin như tập tin cấu hình, tập tin dữ liệu, tập tin nhật ký.

Oracle RAC (Realtime Application Cluster) là một kiến trúc triển khai cơ sở dữ liệu mới của Oracle, nhiều thể hiện của cơ sở dữ liệu có thể kết nối cùng một cơ sở dữ liệu dùng chung hệ thống tập tin lưu trữ. Giúp cho hệ quản trị cơ sở dữ liệu có thể chia tải cho các thể hiện cơ sở dữ liệu, giúp tăng cường khả năng đáp ứng cũng như khả năng phục hồi của cơ sở dữ liệu.

Trên hình chúng ta có 4 thể hiện cơ sở dữ liệu kết nối đồng thời vời một cơ sở dữ liệu bên dưới. Do đó chúng ta có thể kết nối đến bất kỳ thể hiện cơ sỡ dữ liệu nào ở phía trên, thì dữ liệu sẽ được đồng bộ trên 3 thể hiện còn lại.

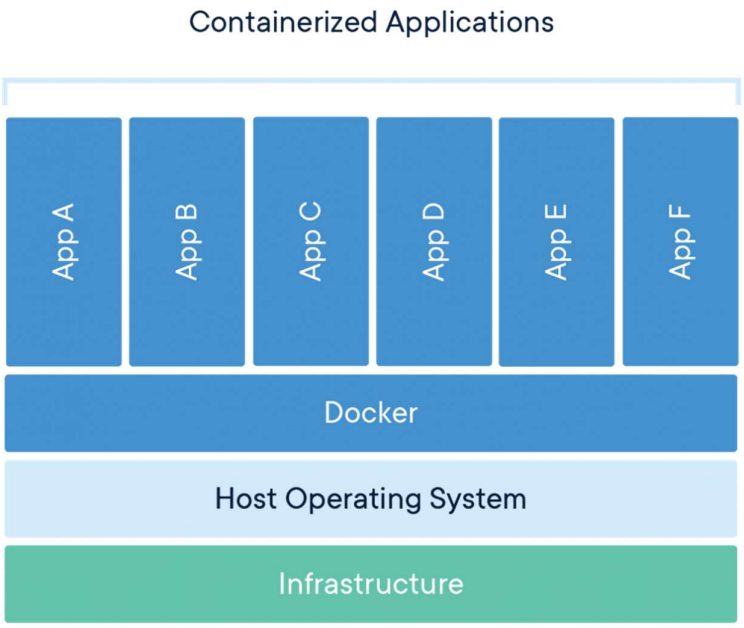


Các thành phần của một Oracle RAC

- Shared Storage
  Là một đĩa lưu trữ dùng chung được chia sẽ cho các thể hiện cơ sở dữ liệu dùng chung, bộ lưu trữ này sẽ được ASM (Automatic Storage Management) quản lý, ASM sử dụng diskgroup để lưu trữ tập tin, một diskgroup bao gồm nhiều đĩa được ASM quản lý như là một đơn vị. Chúng ta có thể thêm hoặc xóa disk trong khi cơ sở dữ liệu tiếp tục truy cập dữ liệu trong group.
- Public Network
  Một mạng dùng chung để cho SCAN(Single Client Access Name) có thể truy cập và chia tải cho các thể hiện cơ sở dữ liệu.
- Private Network
  Một mạng dùng riêng để các thể hiện cơ sở dữ liệu kết nối nội bộ với nhau, giúp đồng bộ dữ liệu nhanh hơn, giúp cho các dữ liệu mới luôn được cập nhật trên tất cả các thể hiện.
- SCAN
  SCAN(Single Client Acces Name) một địa chỉ ip đại diện cho cụm giúp cho phía ứng dụng có thể dễ dàng truy cập vào cụm bằng một địa chỉ, thay vì các địa chỉ ip của thể hiện, việc cài đặt SCAN giúp trên hệ thống tính toán lưới của Oracle RAC giúp cho cụm tự động cân bằng tải giữa các thể hiện.

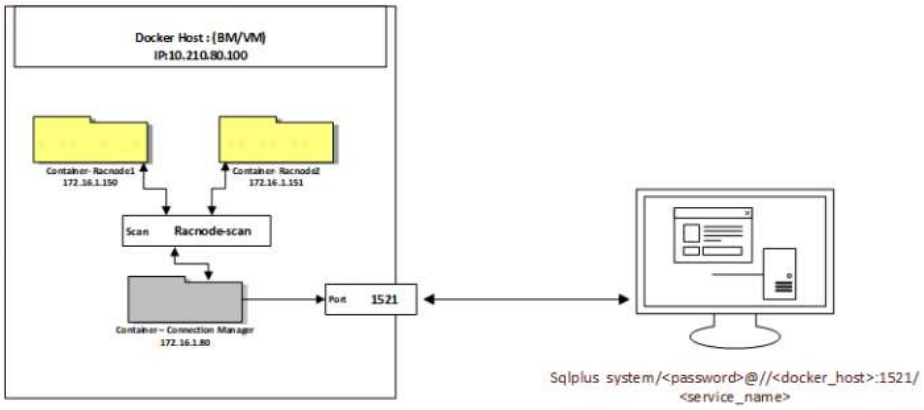
Triển khai kiến trúc Oracle RAC trên nền tảng Docker

- Giới thiệu nền tảng Docker



Docker là một nền tảng mã nguồn mở giúp đóng gói triển khai ứng dụng một cách dễ dàng, người dùng chỉ cần cài đặt các thư viện cần thiết hỗ trợ để chạy ứng dụng của mình sau đó đóng gói nó lại, và có thể di chuyển ứng dụng của mình sau đi đóng gói để triển khai lên các hệ thống khác có chạy Docker bên dưới.

- Triển khai Oracle RAC trên nền tảng Docker



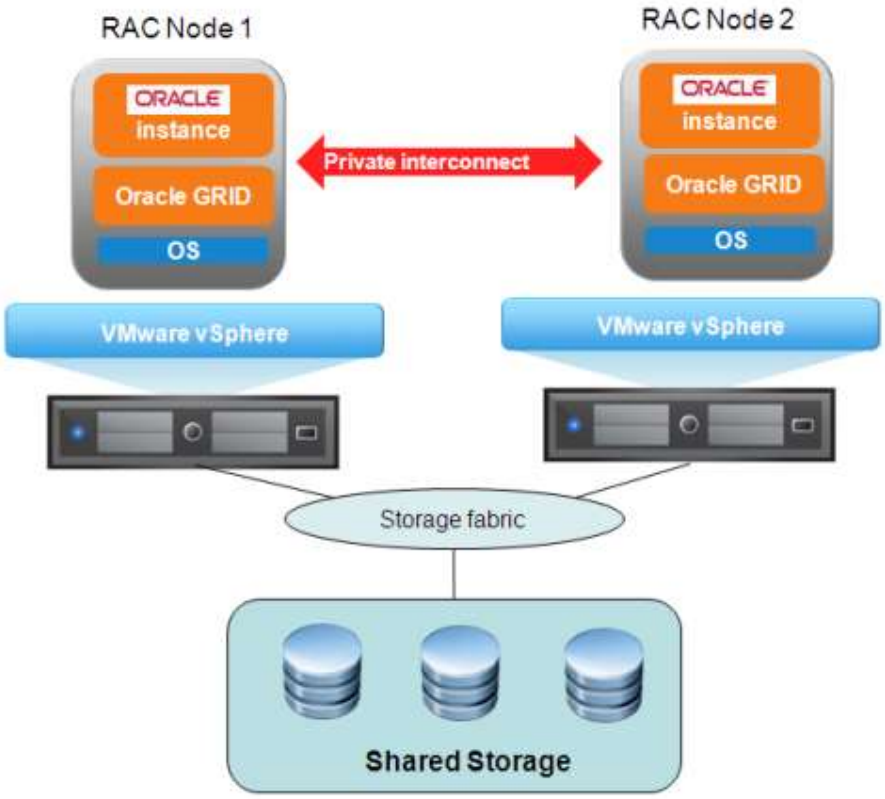
Chúng ta sẽ có tất cả là 3 thành phần:

- Rac storage container
  Container chịu trách nhiệm lưu trữ quản lý data, bên trong container dữ liệu sẽ được oracle ASM quản lý theo diskgroup.

- Scan container
  Container chịu trách nhiệm quản lý tải của cụm, là sẽ export ra một địa chỉ ip duy nhất đại diện cho cụm để phía ứng dụng có thể kết nối vào cụm, và Scan sẽ tự động chia tải cho các racnode container. Nếu không có Scan container thì chúng ta có thể truy cập trực tiếp các racnode container thông qua địa chỉ ip của nó và SID ID riêng của từng racnode, như vậy khi một racnode bị thất bại thì ứng dụng không thể tự động kết nối với một racnode khác được.
- Nodes container
  Các container cài đặt các thể hiện cơ sỡ dữ liệu kết nối vào cùng một container rac-storage.

Triển khai kiến trúc Oracle RAC trên nền tảng máy ảo Vmware

Cũng tương tự như triển khai trên Docker, chúng ta vẫn có các máy ảo dùng để cài đặt các thể hiện cơ sở dữ liệu, chúng ta có thêm một ổ đĩa được chia sẽ dùng chung cho các máy để làm ổ đĩa lưu trữ cơ sở dữ liệu, đĩa này sẽ được Oracle ASM quản lý. Chúng ta có thêm một mạng dùng chung, một mạng dùng riêng để các node giao tiếp bên trong nội bộ.



Triển khai cài đặt máy chủ upload tập tin trên nền tảng Docker Swarm

Docker Swarm là một chế độ cụm của Docker, chúng ta có thể liên kết các máy chủ chạy Docker riêng lẽ lại thành các cụm, giúp tăng khả năng mở rộng của hệ thống, cũng như khả năng chịu lỗi của hệ thống. Mặt định chế độ docker swarm sẽ tự động

cân bằng tải cho hệ thống, ví dụ: chúng ta triển khai 20 server upload file trên một cụm 4 node thì docker swarm sẽ tự động chia mỗi node chạy 5 container. Khi một node nào thất bại thì docker swarm tự động di chuyển 5 container trên node đó sang chia cho 3 node còn lại.
Khả năng dễ dàng mở rộng, chúng ta chỉ cần chỉ định số container của một ứng dụng thì docker swarm sẽ tự động triển khai số lượng container tương ứng trên đều các node. Docker swarm có thể triển khai hàng ngàn server chạy cùng lúc một cách nhanh chóng chỉ với một dòng lệnh ví dụ như:

```
$ docker service scale frontend=50
```

Trong docker swarm thì sẽ có một node làm node master, có nhiệm vụ chia tác vụ cho các node thành viên, khi các node tham gia vào cụm này thì sẽ tham gia vào một mạng chung, gọi là mạng của cụm một node thành viên có thể tham gia hoặc rời khỏi cụm. Nhưng node master không thể rời khỏi cụm nếu node master rời khỏi cụm thì coi như cụm tự hủy.
Nhược điểm của docker swarm là không có cơ chế bầu cử node thành viên trở thành node master nếu như node master thất bại.
Ở đây chúng ta cài đặt cụm gồm 2 node (res80, res81) trong đó res80 làm master, và ta dùng res87 làm máy chủ lưu trữ file. Trên đây ta cài dịch vụ NFS(Network File System) để chia sẻ file cho các container có thể truy cập vào dùng chung. Ta tạo một volume mount vào trong container để lưu trữ file.
Có thể thấy hiện tại có 10 replicated của server upload file.

```
[res80@res80 ~]$ docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE                          PORTS
ofydzmc5tszm        helloworld          replicated          1/1                 alpine:latest
q75uelugcah1        nfs-files-server2   replicated          10/10               lqsang/nfs-files2:latest       *:8080->8080/
tcp
[res80@res80 ~]$
```

Tất cả 10 container này đều mount vào một volume, volume này mount vào máy res87(172.18.54.87) đang chạy dịch vụ nfs.
Chúng ta có thể dễ dàng scacle ứng dụng với lệnh:

```
[res80@res80 ~]$ docker service scale nfs-files-server2=40
nfs-files-server2 scaled to 40
overall progress: 11 out of 40 tasks
```

Và kết quả đạt được:

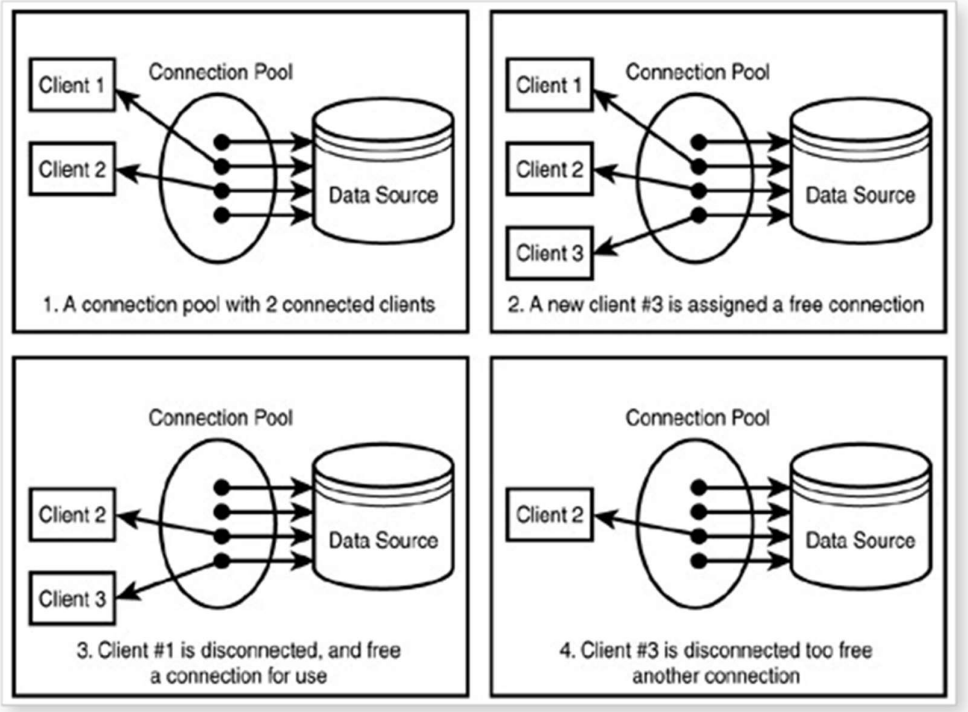
```
[res80@res80 ~]$ docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE                          PORTS
ofydzmc5tszm        helloworld          replicated          1/1                 alpine:latest
q75uelugcah1        nfs-files-server2   replicated          40/40               lqsang/nfs-files2:latest       *:8080->8080/
tcp
[res80@res80 ~]$
```

Như vậy chúng ta có thể thấy docker swarm rất dễ dàng scale out hệ thống. Hiện tại chúng ta có 40 server upload file đang chay, khi một yêu cầu upload file vào api http://172.18.54.80:8080 thì docker swarm sẽ tự động chia tải cho 40 server trong mạng.

IV. ĐÁNH GIÁ KẾT QUẢ

Triển khai được Oracle Rac trên nền tảng docker với 4 node thể hiện,

Giới thiệu Pool connection:



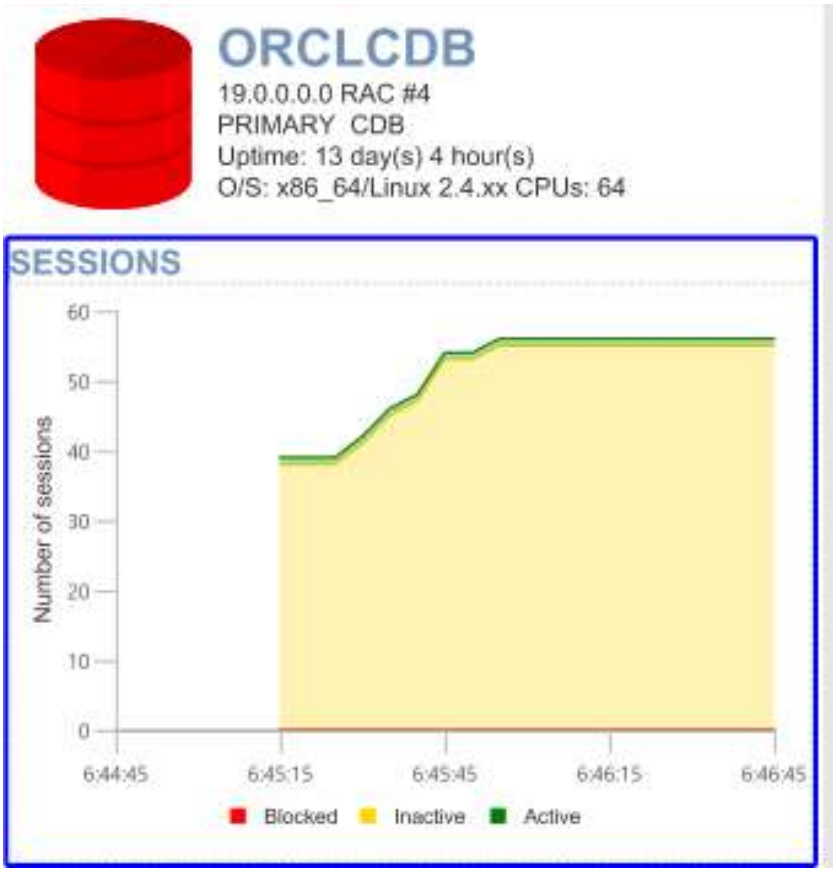
Test trên 1,000,000 yêu cầu vào cụm:

Tiến hành kiểm tra khả năng của hệ thống, dùng phần mềm JMeter để tiến hành kiểm tra tải của cơ sỡ dữ liệu, ở đây chúng ta cấu hình giả lập có 10 nghìn người dùng từ ứng dụng kết nối, tương tác với Oracle RAC của chúng ta, trong phần cấu hình kết nối với cơ sỡ sỡ dữ liệu chúng ta tạo ra một Pool connection với số lượng kết nói duy trì trong Pool là 200 kết nối, mỗi user thực hiện 100 yêu cầu thì tổng cộng chúng ta có 1 triệu yêu cầu đến cơ sỡ dữ liệu. Dưới đây là kết quả sau khi test 1 triệu yêu cầu đến cơ sỡ dữ liệu:

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB...	Sent KB/sec	Avg. Bytes
JDBC Requ...	1000000	4363	5	101384	9856.13	22.43%	1977.4/sec	38.00	0.00	19.7
TOTAL	1000000	4363	5	101384	9856.13	22.43%	1977.4/sec	38.00	0.00	19.7

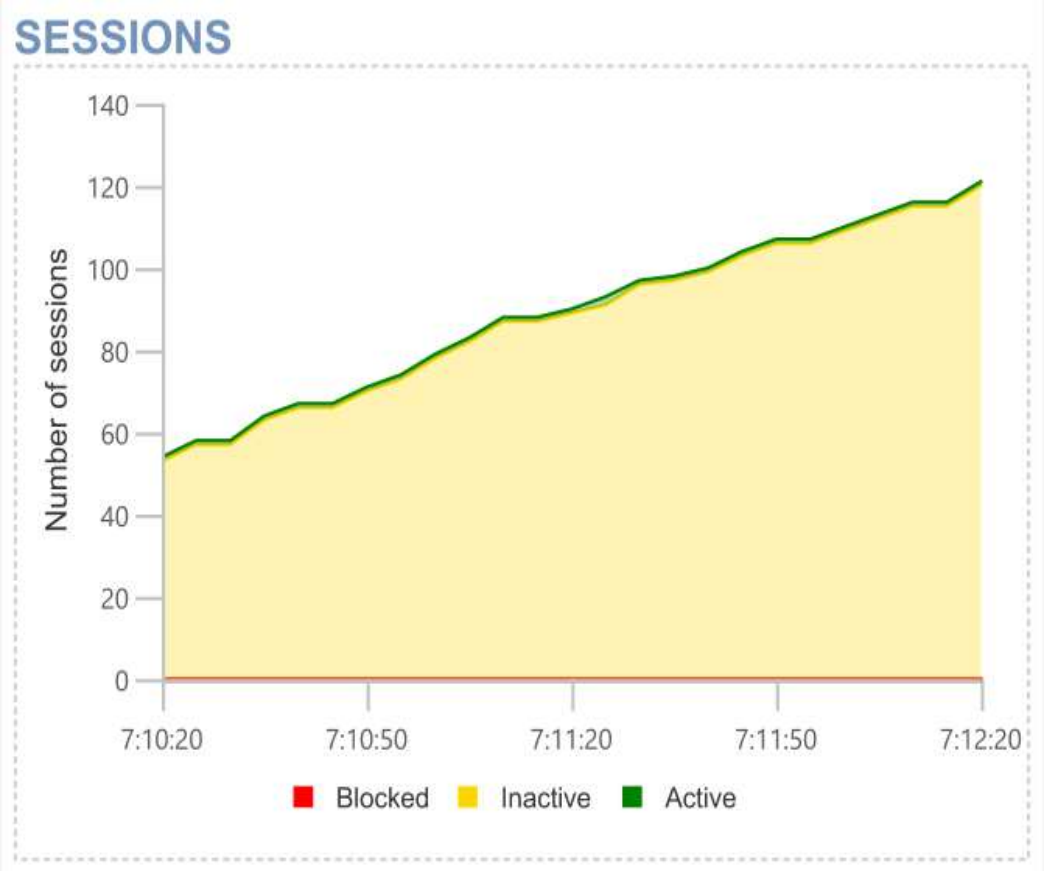
Kế quả trên cho thấy thời gian đáp ứng trung bình trên 1 triệu yêu cần là 4.3 giây, tỷ lệ lỗi gói tin là 22.43% tỉ lệ lỗi ở đây cao là do chúng ta cấu hình số lượng kết nối duy trì trong Pool là 200. Có nghĩa là chúng ta sẽ tạo một đường cho các kết nối vào cơ sỡ dữ liệu mà cụ thể là con đường này nó duy trì 200 kết nối vào, và 200 kết nối này có thể được sử dụng lại bởi các yêu cầu tiếp theo, thay vì mỗi một yêu cầu tạo một kết nối sẽ gây ra quá tải cơ sỡ dữ liệu.

Kế quả cho thấy là với 4 node thì 200 kết nối đồng thời này sẽ được chia tải ra, như vậy mỗi node thì chỉ xử lý khoảng 50-60 kết nối, giúp giảm sự quá tải của cơ sỡ dữ liệu.



Khi chúng ta tinh chỉnh số kết nối trong Pool lên 1000 kết nối thì chúng ta đạt được kết quả tốt hơn cho 1 triệu yêu cầu:

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Through...	Received...	Sent KB/...	Avg. Bytes
JDBC R...	1000000	5186	10	322330	30870.99	0.31%	1872.5/s...	11.32	0.00	6.2
TOTAL	1000000	5186	10	322330	30870.99	0.31%	1872.5/s...	11.32	0.00	6.2



Test trên 200,000 yêu cầu vào cụm:

Và sau đây là một ví dụ khi chúng ta tiến hành test với số lượng user ít hơn:

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB...	Sent KB/sec	Avg. Bytes
JDBC Requ...	200000	994	8	70277	6365.88	0.18%	1813.2/sec	10.81	0.00	6.1
TOTAL	200000	994	8	70277	6365.88	0.18%	1813.2/sec	10.81	0.00	6.1

Dễ dàng nhận thấy khi chúng ta có 2000 users tương tác với hệ thống, thì tỉ lệ kết nối thất bại chỉ 0.18% và thời gian đáp ứng của một yêu cầu trung bình 994ms và thấp nhất là 8ms.

Do các yêu cầu sử dụng một Pool connection để kết nối vào cơ sỡ dữ liệu cho nên số lượng kết nối tối đa vào cơ sỡ dữ liệu chỉ là 200 kết nối như chúng ta đã cấu hình, Số lượng kết nối này phụ thuộc vào việc ta cấu hình trong ứng dụng, tùy vào số kết nối cho phép của cơ sở dữ liệu mà ta đang dùng.

KẾT QUẢ ĐẠT ĐƯỢC QUA ĐỢT THỰC TẬP

- Sau khi thời gian thực tập thực tế tại trung tâm em nhận thấy mình học hỏi được rất nhiều kiến thức mới, cũng như cũng cố lại những kiến thức đã có, những kiến thức được giảng dạy trên trường rất bổ ích, hỗ trợ rất nhiều trong môi trường làm việc thực tế, các kiến thức thực hành được học trên trường rất bổ ích giúp đỡ rất nhiều trong việc tiếp thu các kiến thức mới, cũng như ứng dụng vào thực tiễn.
- Được học tập các kiến thức thực tiễn trong môi trường vận hành máy chủ, cách một hệ thống hoạt động thực sự trong thực tiễn là như thế nào, cách một hệ thống mạng hoạt động.
- Trong quá trình thực tập em cũng đã tiềm hiểu về kiến trúc triển khai cơ sỡ dự liệu phân tán, cũng như là triển khai và chạy thử nghiệm mô hình cơ sỡ dữ liệu phân tán tại trung tâm.
- Triển khai mô hình máy chủ upload tập tin phân tán tại trung tâm làm tiền đề cho quá trình đưa vào dùng thử nghiệm và cài đặt thực tế cho trung tâm.
- Triển khai được mô hình cơ sỡ dữ liệu phân tán trên docker có thể đưa vào thử nghiệm và dùng thử trong thực tế, có thể thay thế hệ thống cơ sỡ dữ liệu hiện tại.