TỔNG QUAN VỀ PROMETHEUS: GIẢI PHÁP GIÁM SÁT HỆ THỐNG TÍCH HỢP VỚI GRAFANA VÀ ỨNG DỤNG TRÊN OPENSTACK

1. Prometheus

1.1. Tổng quan

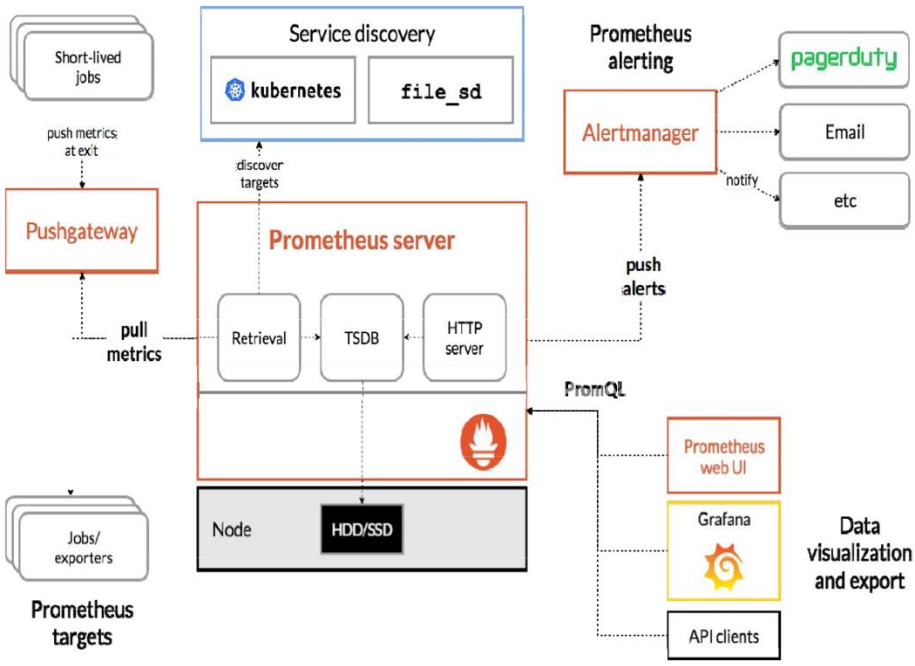
Prometheus là tập hợp các công cụ để giám sát và cảnh báo các hệ thống mã nguồn mở. Promethus được xây dựng bởi SoundCloud và là một dự án mã nguồn mở với cộng đồng người sử dụng và phát triển lớn. Đây là một dự án mã nguồn mở độc lập và đã tham gia vào tổ chức CNCF (Cloud Native Cpmputing Foundation) vào năm 2016, chính thức trở thành thành viên thứ 2 của tổ chức này sau Kubernetes.

Tính năng quan trọng nhất của Prometheus là thu thập thông số, dữ liệu từ các dịch vụ được nhắm đến trong khoảng thời gian cố định đã được cài đặt trước. Ngoài ra, hệ thống được thiết kế để đảm bảo độ tin cậy và có thể hoạt động trong môi trường hệ phân tán. Với ngôn ngữ truy vấn PromQL, Prometheus cung cấp các cách thức linh hoạt để truy xuất và xử lý dữ liệu, giúp phân tích lượng dữ liệu lớn thu thập được cho quá trình đưa ra quyết định.

1.2. Một số thuật ngữ quan trọng

- Time-series Data: là một chuỗi các điểm dữ liệu, thường bao gồm các phép đo liên tiếp được thực hiện từ cùng một nguồn trong một khoảng thời gian.
- Alert : một cảnh báo (alert) là kết quả của việc đạt điều kiện thoả mãn một rule cảnh báo được cấu hình trong Prometheus. Các cảnh báo sẽ được gửi đến dịch vụ Alertmanager.
- Alertmanager: chương trình đảm nhận nhiệm vụ tiếp nhận, xử lý các hoạt động cảnh báo.
- Client Library: một số thư viện hỗ trợ người dùng có thể tự tuỳ chỉnh lập trình phương thức riêng để lấy dữ liệu từ hệ thống và đẩy dữ liệu metric về Prometheus.
- Endpoint: nguồn dữ liệu của các chỉ số (metric) mà Prometheus sẽ đi lấy thông tin.
- Exporter: là một chương trình được sử dụng với mục đích thu thập, chuyển đổi các metric không ở dạng kiểu dữ liệu chuẩn Prometheus sang chuẩn dữ liệu Prometheus. Sau đấy exporter sẽ expose web service api chứa thông tin các metrics hoặc đẩy về Prometheus.
- Instance: một instance là một nhãn (label) dùng để định danh duy nhất cho một target trong một job .
- Job: là một tập hợp các target chung một nhóm mục đích. Ví dụ: giám sát một nhóm các dịch vụ database,… thì ta gọi đó là một job .
- PromQL: là viết tắt của Prometheus Query Language, ngôn ngữ này cho phép bạn thực hiện các hoạt động liên quan đến dữ liệu metric.
- Sample: là một giá trị đơn lẻ tại một thời điểm thời gian trong khoảng thời gian time series.
- Target: một target là định nghĩa một đối tượng sẽ được Prometheus đi lấy dữ liệu (scrape). Ví dụ như: nhãn nào sẽ được sử dụng cho đối tượng, hình thức chứng thực nào sử dụng hoặc các thông tin cần thiết để quá trình đi lấy dữ liệu ở đối tượng được diễn ra.

1.3. Kiến trúc Prometheus



Hình 1. Kiến trúc Prometheus

Kiến trúc trọng tâm của Prometheus được biểu diễn ở hình 1, các mô-đun chính của Prometheus bao gồm 6 thành phần:

- Promethus server: lấy dữ liệu từ các targets đã được cấu hình hoặc từ service discovery như DNS, consul, K8s, meos,..
- Exporters: hỗ trợ giám sát các dịch vụ hệ thống và gửi về Prometheus các dữ liệu theo chuẩn mong muốn
- Push Gateway: sử dụng để hỗ trợ các job có thời gian thực hiện ngắn (do các job này tồn tại không đủ lâu để Prometheus chủ động lấy dữ liệu nên các dữ liệu từ các công việc này sẽ được đẩy về Push Gateway rồi đẩy về Prometheus)
- PromQL (Promethéu Query Language): ngôn ngữ cho phép thực hiện truy vấn với dữ liệu metric (thông số)
- Alertmanager: dịch vụ quản lý, xử lý các cảnh báo
- Web UI: biểu diễn dữ liệu

Prometheus scrape các metric từ các job/exporter hoặc từ pushgateway. Nó sẽ lưu trữ data và chạy các rule qua các data này để tổng hợp và ghi lại time series mới từ dữ liệu đã tồn tại hoặc là tạo cảnh báo. Grafana hoặc các API consumer có thể sử dụng để vẽ những dữ liệu thu thập được.

1.4. Prometheus metrics

Prometheus giám sát các điểm đầu cuối (endpoints) và cung cấp 4 loại thông số (metrics):

1.4.1. Counter (Bộ đếm)

Loại dữ liệu tích lũy này phù hợp để theo dõi số lượng yêu cầu, lỗi hoặc tác vụ hoàn thành. Nó không thể giảm, chỉ có thể tăng lên hoặc được đặt lại về 0.

Counter nên sử dụng trong các trường hợp:

- Ghi lại giá trị chỉ tăng
- Đánh giá tốc độ tăng

Các trường hợp sử dụng cho counter bao gồm số lượng yêu cầu, số tác vụ hoàn thành và số lỗi.

1.4.2. Gauge (Đồng hồ)

Loại dữ liệu theo thời gian thực này có thể tăng hoặc giảm. Nó phù hợp để đo lường mức sử dụng bộ nhớ hiện tại và các yêu cầu đồng thời.

Nên sử dụng Gauge để:

- Ghi lại giá trị có thể tăng hoặc giảm
- Các trường hợp không cần truy vấn tốc độ của giá trị

Các trường hợp sử dụng cho gauge bao gồm kích thước hàng đợi, mức sử dụng bộ nhớ và số lượng yêu cầu đang được xử lý.

1.4.3. Histogram (Biểu dồ phân bố)

Loại dữ liệu này phù hợp cho các phép đo tổng hợp, bao gồm thời gian yêu cầu, kích thước phản hồi và điểm Apdex đo lường hiệu suất ứng dụng. Biểu đồ phân bố lấy mẫu quan sát và phân loại dữ liệu vào các nhóm mà bạn có thể tùy chỉnh.

Nên sử dụng Histogram để:

- Đo nhiều lần một giá trị duy nhất, cho phép tính trung bình hoặc phần trăm
- Các giá trị có thể xấp xỉ
- Một dải giá trị mà bạn xác định trước, bằng cách sử dụng các định nghĩa mặc định trong nhóm biểu đồ phân bố hoặc các giá trị tùy chỉnh của bạn

Các trường hợp sử dụng cho histogram bao gồm thời gian yêu cầu và kích thước phản hồi.

1.4.4. Summary

Loại dữ liệu này phù hợp để đo lường chính xác các quartile (phần tư). Summary lấy mẫu quan sát và cung cấp tổng số quan sát, tổng các giá trị quan sát được và tính toán các quartile.

Nên sử dụng Summary để:

- Đo nhiều lần một giá trị duy nhất, cho phép tính trung bình hoặc phần trăm
- Các giá trị có thể xấp xỉ
- Một dải giá trị mà bạn không thể xác định trước, do đó biểu đồ phân bố không phù hợp

Các trường hợp sử dụng cho summary bao gồm thời gian yêu cầu và kích thước phản hồi.

1.5. Prometheus advantages & disadvantages

2. Prometheus Grafana

2.1. Grafana introduction

Grafana là một nền tảng mã nguồn mở dành cho việc giám sát và trực quan hóa dữ liệu quan sát. Giao diện người dùng đẹp mắt và thân thiện của nó cho phép người dùng thu thập dữ liệu từ nhiều nguồn khác nhau, bao gồm cả các số liệu từ Prometheus, để cung cấp một cái nhìn toàn diện về dữ liệu.

Điểm mạnh của Grafana nằm ở khả năng tạo các bảng điều khiển (dashboard) tương tác, phong phú, hiển thị các số liệu thông qua biểu đồ, đồ họa và cảnh báo. Những hình ảnh trực quan này có thể được tùy chỉnh và mở rộng bằng nhiều plugin khác nhau, hỗ trợ các nguồn dữ liệu bổ sung và các kiểu bảng điều khiển khác nhau, cho phép người dùng xem và phân tích dữ liệu theo thời gian thực.

Grafana có 6 tính năng chính dưới đây:

2.1.1. Display mode

- Biểu đồ thân thiện, linh hoạt cho người dùng.
- Plugin bảng điều khiển cung cấp nhiều cách thức trực quan hóa các chỉ số và nhật ký khác nhau.
- Thư viện chính thức có nhiều plugin bảng điều khiển đa dạng, chẳng hạn như bản đồ nhiệt, biểu đồ đường, biểu đồ hình tròn và các phương thức hiển thị khác

2.1.2. Data source

Nguồn dữ liệu bao gồm Graphite, InfluxDB, OpenTSDB, Prometheus, Elasticsearch, CloudWatch và KairosDB.

2.1.3. Notice reminder

Grafana cho phép xác định trực quan các quy tắc cảnh báo cho các chỉ số quan trọng nhất. Grafana sẽ liên tục tính toán và gửi thông báo, thông qua Slack, PagerDuty,... khi dữ liệu đạt đến ngưỡng nhất định.

2.1.4. Mixed Display

Grafana có thể kết hợp các nguồn dữ liệu khác nhau trong cùng một biểu đồ và có thể chỉ định hoặc thậm chí tùy chỉnh nguồn dữ liệu dựa trên từng truy vấn.

2.1.5. Annotation

Grafana có khả năng chú thích biểu đồ từ các nguồn dữ liệu khác nhau. Di chuột qua một sự kiện sẽ hiển thị đầy đủ siêu dữ liệu và thẻ của sự kiện đó.

2.1.6. Filter

Bộ lọc Ad-hoc cho phép tạo động các bộ lọc key/value mới, tự động áp dụng cho tất cả các truy vấn sử dụng nguồn dữ liệu.

2.2. Prometheus và Grafana trong môi trường Cloud Native

- Khả năng mở rộng và linh hoạt: Cơ sở dữ liệu chuỗi thời gian hiệu quả của Prometheus và khả năng truy vấn và trực quan hóa dữ liệu từ nhiều nguồn của Grafana làm cho chúng đặc biệt phù hợp với các môi trường đám mây động. Khả năng thích ứng này đảm bảo rằng khi hệ thống của bạn phát triển hoặc thay đổi, khả năng giám sát và trực quan hóa của bạn có thể phát triển theo mà không cần cấu hình lại đáng kể.

- Khả năng quan sát nâng cao: Khả năng thu thập nhiều loại số liệu từ các nguồn khác nhau của Prometheus, kết hợp với khả năng trực quan hóa mạnh mẽ của Grafana, mang lại cái nhìn toàn diện về tình trạng sức khỏe của hệ thống. Điều này bao gồm mọi thứ từ việc sử dụng CPU và bộ nhớ đến các số liệu ứng dụng tùy chỉnh, cho phép có được những thông tin chi tiết chi tiết về hiệu suất ở cả mức độ hệ thống và mức độ vi mô.

- Giải pháp giám sát tiết kiệm chi phí: Cả Prometheus và Grafana đều là các công cụ mã nguồn mở, loại bỏ nhu cầu về phần mềm độc quyền đắt tiền trong khi vẫn cung cấp các khả năng giám sát và trực quan hóa mạnh mẽ.

- Giải quyết vấn đề và cảnh báo chủ động: Prometheus có thể được cấu hình để kích hoạt cảnh báo dựa trên các số liệu hoặc điều kiện cụ thể, sau đó được hiển thị trong các bảng điều khiển của Grafana. Điều này cho phép các nhóm nhận dạng và giải quyết các vấn đề tiềm ẩn trước khi chúng ảnh hưởng đến người dùng, cải thiện thời gian hoạt động và độ tin cậy của hệ thống.

2.3. Ứng dụng của Prometheus và Grafana

2.3.1. Giám sát Kubernetes

Kubernetes đã trở thành nền móng cho các kiến trúc microservices, làm cho việc giám sát các thành phần và tài nguyên của nó trở nên quan trọng cho hiệu quả hoạt động của quá trình vận hành. Bằng cách tích hợp Prometheus với Grafana, ta có thể nắm bắt tình trạng và hiệu suất của các cụm Kubernetes trong thời gian thực.

Prometheus tích hợp với Kubernetes và xuất sắc trong việc thu thập các số liệu liên quan đến hiệu suất cụm Kubernetes, chẳng hạn như việc sử dụng CPU và bộ nhớ của nút, thống kê pod và lưu lượng mạng. Bằng cách thiết lập Prometheus để quét các số liệu từ Kubernetes API, cAdvisor và kubelet, ta có thể giám sát việc tiêu thụ và yêu cầu tài nguyên trên toàn cụm của mình.

Grafana sau đó có thể trực quan hóa các số liệu này, giúp bạn xác định các điểm nút thắt cổ chai hoặc tài nguyên chưa được sử dụng hết. Điều này cho phép bạn cân bằng tải hiệu quả hơn và tối ưu hóa cấu hình cụm của mình để có hiệu suất tốt hơn.

Ngoài các số liệu hạ tầng, Prometheus và Grafana có thể giám sát tình trạng và hiệu suất của các ứng dụng chạy trên Kubernetes. Prometheus có thể thu thập các số liệu tùy chỉnh do ứng dụng của bạn cung cấp bằng cách sử dụng các thư viện khách hàng, bao gồm bất kỳ thứ gì từ độ trễ đến KPI kinh doanh. Các bảng điều khiển Grafana (Grafana dashboards) sau đó có thể hiển thị các số liệu này, cho phép bạn theo dõi tình trạng sức khỏe ứng dụng, thời gian phản hồi và thông lượng.

2.3.2. Điều chỉnh và tối ưu hoá hiệu suất ứng dụng

Với Prometheus thu thập các số liệu chi tiết về việc vận hành của hệ thống, bạn có thể xác định các khu vực mà hiệu suất ứng dụng có thể không đạt tiêu chuẩn. Có thể các truy vấn nhất định mất quá nhiều thời gian để thực hiện, hoặc việc sử dụng bộ nhớ liên tục cao. Prometheus cung cấp dữ liệu cần thiết để xác định các điểm không hiệu quả này.

Khi đã thu thập được dữ liệu cần thiết, Grafana sẽ giúp trực quan hóa các dữ liệu này. Bằng cách tạo các dashboards phù hợp với từng nhu cầu cụ thể, giúp ta có thể quan sát các số liệu hiệu suất theo thời gian thực hoặc xem lại dữ liệu trong 1 khung thời gian để phát hiện xu hướng. Điều này có thể hỗ trợ đưa ra các quyết định về nơi phân bổ tài nguyên, khi nào cần mở rộng hạ tầng và cách điều chỉnh cấu hình để đạt hiệu suất tối ưu.

2.3.3. Cảnh báo và phát hiện bất thường

Trong bất kỳ môi trường công nghệ nào, các bất thường và hành vi không mong đợi có thể là tiền đề của các vấn đề lớn hơn. Với các quy tắc cảnh báo của Prometheus, ta có thể định nghĩa các điều kiện mà khi được đáp ứng sẽ kích hoạt một cảnh báo. Cách tiếp cận chủ động này đảm bảo ta có thể nhận thức được các vấn đề tiềm ẩn trước khi chúng gây ra downtime hoặc làm hiệu suất dịch vụ suy giảm.

Grafana phát triển cách tiếp cận này bằng cách cung cấp bối giao diện trực quan cho các cảnh báo này. Khi một cảnh báo được kích hoạt, người giám sát có thể nhanh chóng chuyển đến bảng điều khiển tương ứng để đánh giá tình hình. Việc truy cập ngay lập tức vào dữ liệu trực quan giúp phán đoán được nguyên nhân gốc rễ và tìm ra cách xử lý phù hợp.

Ngoài ra, Grafana có thể được cấu hình để gửi thông báo qua các kênh khác nhau, đảm bảo rằng những người có liên quan được thông báo ngay lập tức khi một vấn đề được phát hiện.

2.3.4. Trực quan hoá chi phí đám mây

Với sự bùng nổ của điện toán đám mây, việc quản lý chi phí đã trở thành một phần quan trọng của quá trình vận hành các dự án công nghệ thông tin. Prometheus có thể được cấu hình để thu thập dữ liệu liên quan đến việc sử dụng tài nguyên đám mây, là bước quan trọng hướng tới việc kiểm soát và tối ưu hóa các loại chi phí. Các số liệu này có thể bao

gồm số lượng phiên bản đang chạy, dung lượng lưu trữ sử dụng và hoạt động truyền tải mạng,...

Grafana sau đó có thể lấy dữ liệu này và trực quan hoá thành các dashboards hiển thị các xu hướng chi tiêu cho các dịch vụ điện toán đám mây. Các bảng điều khiển này có thể được tùy chỉnh để hiển thị thông tin hữu ích nhất cho việc giám sát tài chính, chẳng hạn như chi phí theo phòng ban, dự án hoặc dịch vụ cụ thể. Bằng cách có một biểu đồ trực quan về chi phí đám mây, người giám sát có thể phát hiện các điểm không hiệu quả và thực hiện thay đổi để giảm bớt các chi phí không cần thiết.

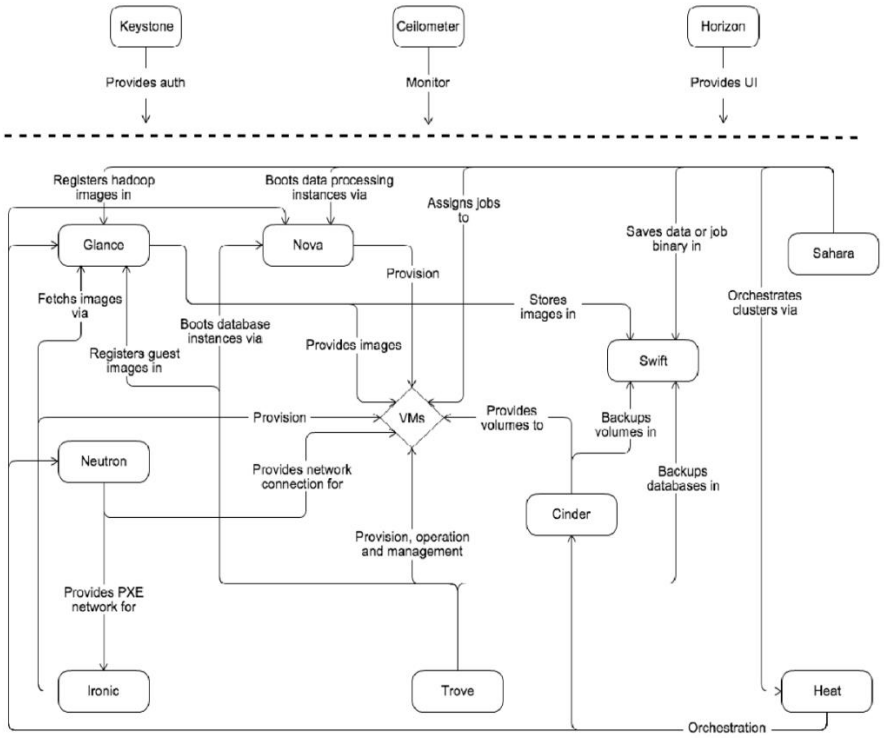
2.3.5. Theo dõi SLI (Service Level Indicators) và SLO (Service Level Objectives)
Các Chỉ số mức dịch vụ (SLI) và Mục tiêu mức dịch vụ (SLO) rất quan trọng trong việc đo lường độ tin cậy của dịch vụ. Prometheus chuyên thu thập các số liệu đóng vai trò là SLI, cung cấp một bức tranh chi tiết về mức độ hiệu suất của dịch vụ so với các tiêu chuẩn đã được định nghĩa từ trước.
Grafana sau đó có thể giúp theo dõi tiến độ trong việc đạt các SLO này. Bằng cách tạo các bảng điều khiển tập trung vào các chỉ số hiệu suất chính, người giám sát có thể liên tục giám sát sự tuân thủ với các SLOs. Vòng phản hồi liên tục này cho phép người giám sát thực hiện các điều chỉnh cần thiết để đảm bảo rằng mức dịch vụ nằm trong các ngưỡng chấp nhận được.

3. Monitoring system of OpenStack Cloud Platform based on Prometheus

3.1. OpenStack introduction

OpenStack ban đầu bao gồm Nova được phát triển. bởi NASA và Swift. được phát triển bởi Rackspace , sau đó được cấp phép bởi Âpche. Đây là một dự án mã nguồn mở để xây dựng các public và private clouds cũng như quản lý nền tảng đám mây (cloud platform).Với sự tham gia của các nhà sản xuất máy chủ, các nhà phân phối Linux, các nhà sản xuất thiết bị mạng, các nhà sản xuất lưu trữ và các gã khổng lồ trong lĩnh vực IT, cộng đồng OpenStack hoạt động sôi nổi hơn tất cả các đối thủ cạnh tranh, và các dự án mới tiếp tục phát triển, trở thành nền tảng đám mây có mã nguồn mở được sử dụng rộng rãi nhất.

OpenStack tách người dùng khỏi các tài nguyên phần cứng phong phú đằng sau thông qua mạng lưới. Một mặt, OpenStack chịu trách nhiệm tương tác với Hypervisor chạy trên nút vật lý để thực hiện quản lý và kiểm soát các tài nguyên phần cứng khác nhau, mặt khác thì cung cấp cho người dùng một máy ảo để đáp ứng nhu cầu của họ. Cho đến nay, OpenStack đã tích hợp 10 thành phần. Các thành phần này bao gồm dịch vụ tính toán Nova, dịch vụ mạng Neutron, dịch vụ lưu trữ đối tượng Swift, dịch vụ lưu trữ khối Cinder, dịch vụ hình ảnh Glance, dịch vụ xác thực Keystone, dịch vụ giao diện người dùng Horizon, dịch vụ giám sát Ceilometer, dịch vụ điều phối Heat, và dịch vụ cơ sở dữ liệu Trove. Mối quan hệ giữa các thành phần được thể hiện trong Hình 2.

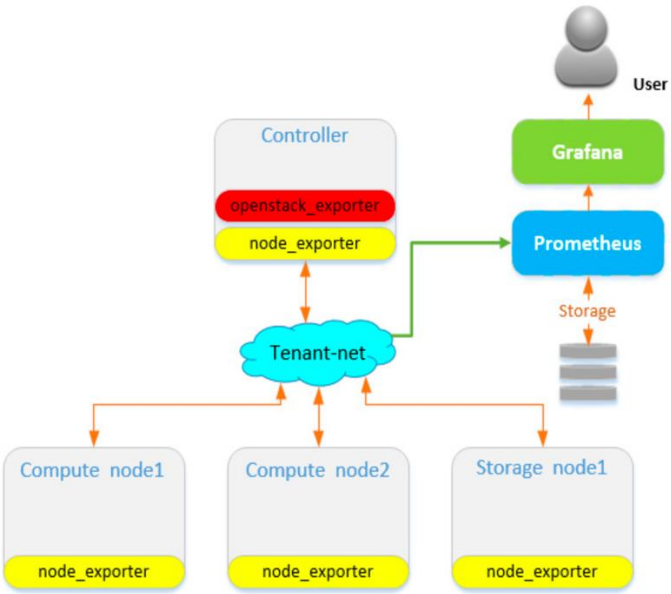


Hình 2. Kiến trúc OpenStack

3.2. System design

3.2.1. System structure

Hệ thống này thực hiện việc thu thập dữ liệu hệ thống của từng nút bằng cách triển khai node_exporter trong mỗi nút của nền tảng đám mây OpenStack, và thu thập dữ liệu của toàn bộ nền tảng đám mây OpenStack bằng cách triển khai openstack_exporter trong nút điều khiển. Hệ thống truyền dữ liệu được thu thập bởi các exporters đến Prometheus, nơi thu thập và lưu trữ dữ liệu, Grafana sẽ đọc dữ liệu, sau đó tích hợp dữ liệu để hiển thị trực quan. Sơ đồ cấu trúc hệ thống được thể hiện trong Hình 3.



Hình 3. Kiến trúc hệ thống

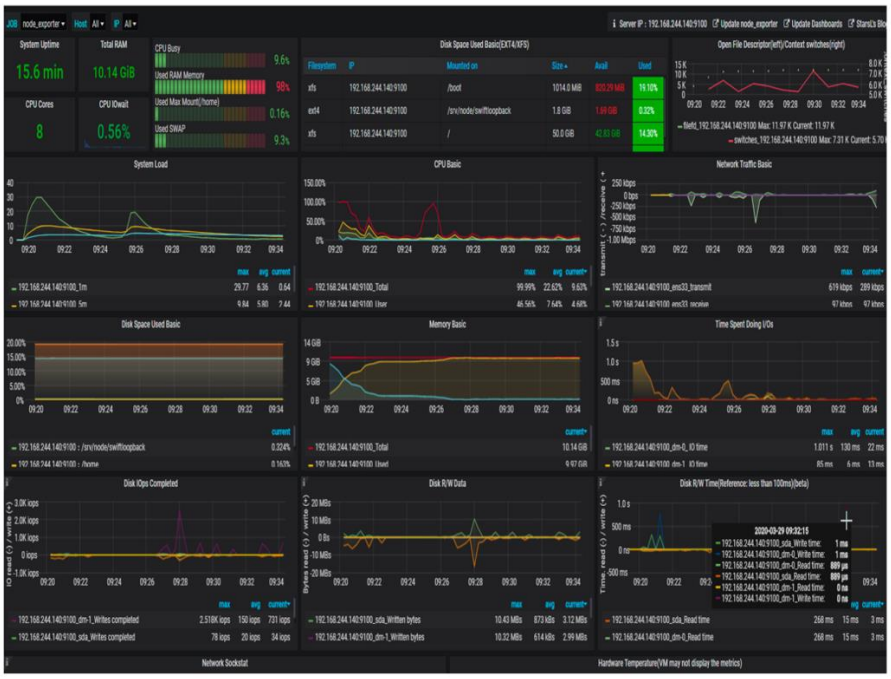
3.2.2. Node_exporter

Node _exporter chủ yếu thu thập các thông tin của các nút máy chủ, như server CPU, bộ nhớ, disk, I/O,... Các tham số chính của Node_exporter được thể hiện ở Bảng 1.

Name	Function
Arp	Collect ARP statistics
CPU	Collect CPU statistics
Diskstats	Collect disk I/O statistics
Filesystem	Collect documentation system statistics, such as disk used space
Loadavg	Collect system load information
Meminfo	Collect memory statistics
Netdev	Collect the net mouth flow statistics
Netsat	Collect network statistics
Stat	Collect various system statistics, including system startup time, forks, interrupts, etc
Uname	Gets the system information through the uname system call

Bảng 1. Node_exporter parameter function comparison

Grafana đọc các dữ liệu thu thập được từ node_exporter của Prometheus và hiển thị biểu đồ trực quan sau quá trình xử lý tích hợp. Ví dụ về các biểu đồ trực quan thể hiện ở Hình 4.



Hình 4. Node monitoring display

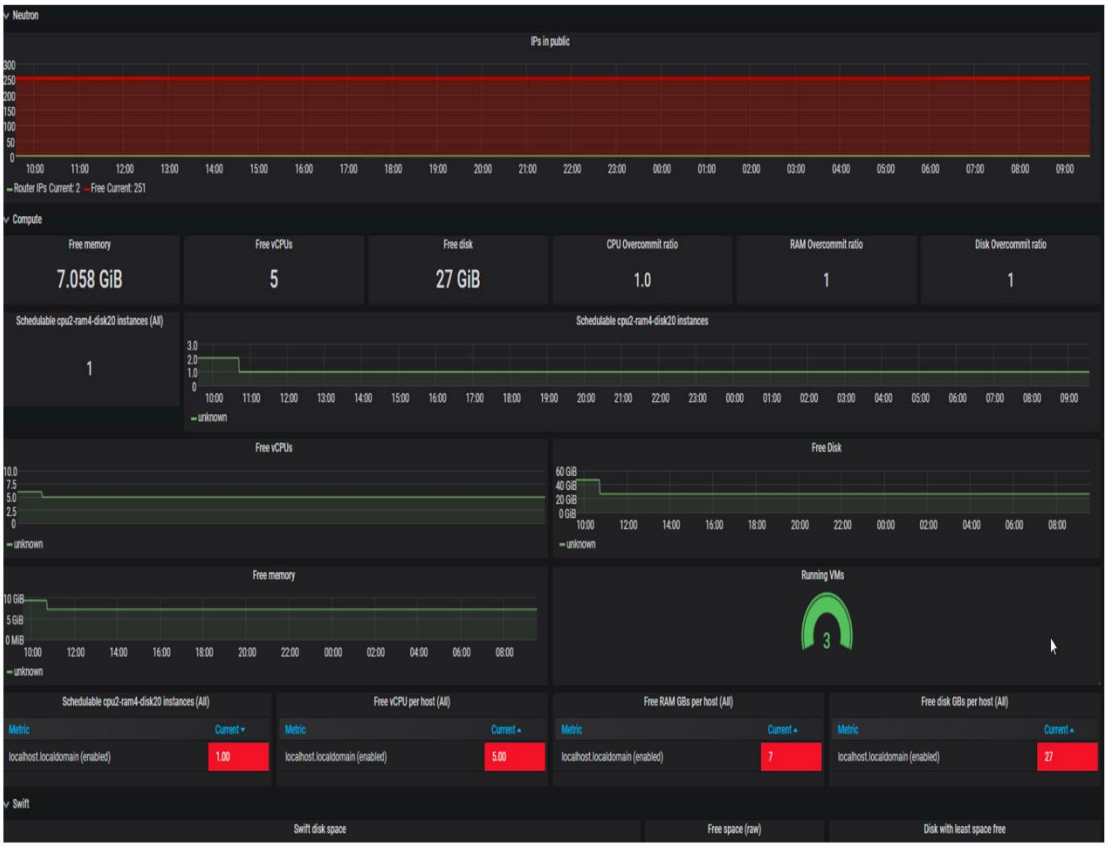
3.2.3. OpenStack_exporter

OpenStack_exporter chủ yếu thu thập dữ liệu cơ bản của nền tảng đám mây OpenStack như bộ nhớ, vcpus, disk, số lượng phiên bản (instance), mạng,.. Các tham số chính của OpenStack_exporter được biểu diễn ở bảng 2.

Name	Function
Meminfo	Collect memory statistics
VCPUs	Collect VCPUs statistics
RAM	Collect RAM statistics
CPU	Collect CPU statistics
Disk	Collect CPU statistics, such as disk free space, disk utilization
Instances	Collect virtual machines statistics, such as the number of virtual machines
Swift	Collect object storage statistics
Neutron	Collect network statistics

Bảng 2. OpenStack exporter parameter function comparison

Grafana đọc các dữ liệu thu thập được từ OpenStack_exporter của Prometheus và hiển thị biểu đồ trực quan sau quá trình xử lý tích hợp. Ví dụ về các biểu đồ trực quan thể hiện ở Hình 5.



Hình 5. OpenStack cloud platform monitoring

3.3. Kết luận

Kết hợp với các đặc điểm của nền tảng đám mây OpenStack, bài báo [1] thiết kế và triển khai một hệ thống giám sát toàn diện, thông minh và hiệu quả bằng cách sử dụng công cụ giám sát mạnh mẽ Prometheus và công cụ trực quan hóa phong phú Grafana. Thí nghiệm chứng minh rằng hệ thống có hiệu quả và khả năng hoạt động theo thời gian thực tốt, giúp nền tảng đám mây OpenStack hoàn thiện hơn và cũng giúp duy trì độ tin cậy và ổn định của nền tảng đám mây OpenStack.

4. Tham khảo

[1]: L. Chen, M. Xian and J. Liu, "Monitoring System of OpenStack Cloud Platform Based on Prometheus," 2020 International Conference on Computer Vision, Image and Deep Learning (CVIDL), Chongqing, China, 2020

[2]: Prometheus Monitoring: Use Cases, Metrics, and Best Practices

[3] Prometheus Documentation