**Báo cáo sơ bộ**

Phan Công Huân

**1. Đặt vấn đề**

**1.1 Đưa ra bài toán**

- Hệ thống quản lý và trích xuất dữ liệu từ IoT có 2 tầng. Tầng Fog là nơi triển khai, theo dõi trạng thái các thiết bị IoT (sensor, gateway, platform). Tầng Cloud là nơi tiếp nhận dữ liệu về trạng thái theo dõi được của các thiết bị, dữ liệu từ các sensor thu thập được từ tầng Fog.

Các sensor triển khai trên tầng Fog được quản lý bởi các IoT platform, ta gọi đó là các task. Ngoài ra ở tầng này còn triển một số task khác như bộ lọc dữ liệu từ các IoT platform, các task mining dữ liệu thu được từ sensor.. Tất cả các task này được triển khai trên một cơ sở hạ tầng là các Node server. Các node server này hữu hạn về mặt số lượng, mặt tài nguyên vật lý của từng node.

- Vì vậy để đạt được các mục tiêu tối đa hóa các task được xử lý và giảm thiểu hóa tài nguyên (giảm chi phí) phải sử dụng trên các node (giảm số node sử dụng), trong bối cảnh các task biến thiên liên tục và các node là hữu hạn, ta cần phải có một hệ thống scheduler để đạt được các mục tiêu trên.

**1.2 Mục đích của đồ án**

Xây dựng hệ thống scheduler nhằm thực hiện các mục tiêu trong bối cảnh:

- *Mục tiêu*:

+ Tối đa hóa các task được xử lý

+ Giảm thiểu hóa tài nguyên phải sử dụng trên các node (giảm số lượng node sử dụng)

- *Bối cảnh*:

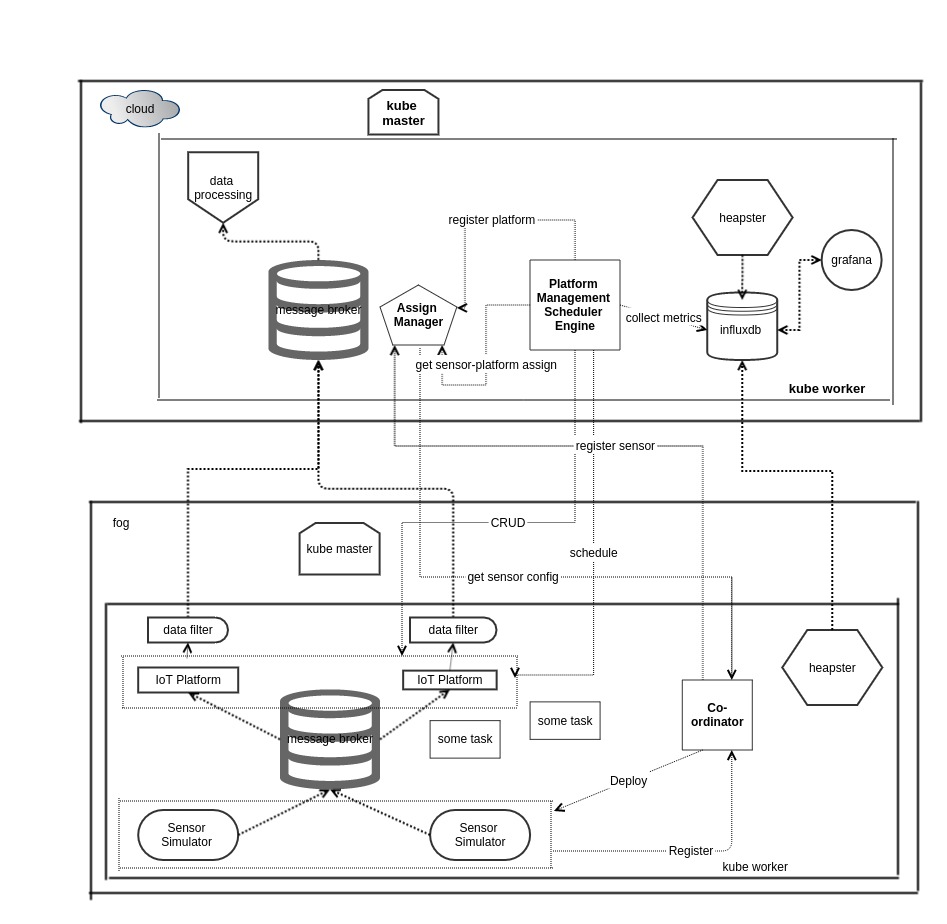
+ Tài nguyên tầng fog là hữu hạn (các node)

+ Các task được tạo mới và thay thế liên tục

+ Các thiết bị sensor biến đổi liên tục (thêm mới, thay thế, mất kết nối…)

+ Các kết nối giữa các thành phần trong tầng fog có nhiều biến động

**2. Kiến trúc hệ thống**

**2.1 Các độ đo (metric) cần thu thập**

- *Sensor simulator*:

+ type

+ quantity

+ runtime parameters: data rate, data range, distribution (?)

+ id

- *IoT Platform*:

+ relative metric: RAM, CPU, Network

- *Node*:

+ relative metric: RAM, CPU, Network

**2.2 Các thành phần quản lý**

**2.2.1 Sensor Co-ordinator**

- CRUD sensor

- Lưu trữ số lượng, vị trí, config, trạng thái (in, out) của các sensor

**2.2.2 Platform Management**

- Engine scheduling: assign platform này chạy trên node nào, tăng giảm platform …

+ Input: các metrics từ node, platform như trong phần 1.1

+ Output:

++ Số lượng các platform replica

++ platform instance được assign vào node

- CRUD platform

- Collect metric từ influxdb, analyze monitor data từ các platform ở tầng fog

- Lấy sensor config từ **Assigment Manager**

**2.2.3 Assigment Manager**

- Quản lý việc một platform instance sẽ nhận dữ liệu từ sensor nào

- Lưu trữ, quản lý các thông tin về mối liên hệ giữa platform với sensor

- Thực hiện việc phân tải dữ liệu sensor cho các platform theo 2 cách:

+ First in first server

+ Round robin

**3. Một số vấn đề gặp phải trong quá trình triển khai hệ thống**

3.1 Hiện tại Task mới chỉ có một loại là IoT platform, và platform mới chỉ sử dụng một loại là OpenHAB

3.2 Các Task được triển khai trên các Docker container, và tài nguyên tiêu thụ của các container này là không có giới hạn. Điều này nảy sinh vấn đề cần thiết lập được một định mức tài nguyên cho 1 Task là bao nhiêu để có mức tham chiếu cho việc thực hiện được cơ chế replacement.

Ví dụ: Node A còn thừa 10% RAM, node B còn thừa 5% RAM, cần deploy một Task C vào 1 trong 2 node này nhưng không biết Task C chiếm bao nhiêu % RAM đê thực hiện ra quyết định.

3.3 Cơ chế phân tải dữ liệu tiếp nhận từ các sensor cho các platform. Tài nguyên tiêu thụ của một platform thay đổi khi lượng dữ liệu đổ về từ sensor thay đổi, vì vậy cần cơ chế phân tải hợp lý để các platform instance không tiêu thụ quá giới hạn tài nguyên của node.

3.4 Tài nguyên tiêu thụ trên các platform (CPU, Network) biến thiên tuyến tính theo lượng dữ liệu đổ về từ sensor, Ram thì tăng dần theo quá trình hoạt động của platform.

N: số sensor gửi dữ liệu, f=tần suất gửi dữ liệu của 1 sensor (số lần/s)

+

+ Giải thích

++ Khi lượng dữ liệu đổ về platform tăng dẫn đến lượng tiêu thụ tài nguyên CPU, Network của platform tăng gần đúng theo một hàm tuyến tính.

++ Lượng tiêu thụ RAM của platform tăng dần theo thời gian sử dụng của platform đến một ngưỡng nào đó thì không đổi (ổn định). --> sẽ lấy ngưỡng ổn định làm mức tiêu thụ tài nguyên RAM.

+ Kết luận

++ Có thể sử dụng những phân tích dữ liệu trên làm cơ sở cho việc xác định mức tài nguyên tiêu thụ của platform/ task (vấn đề 3.2).

++ Làm cơ sở để giải quyết bài toán phân tải dữ liệu cho các platform.

**4. Một số hướng phát triển sắp tới cho bài toán**

**4.1 Real scheduling**

**4.1.1 Xây dựng tập luật cho Engine scheduling**

- Mục tiêu

+ Tối đa hóa các task được xử lý

+ Giảm thiểu hóa tài nguyên phải sử dụng trên các node (giảm số lượng node sử dụng).

- Các luật

*1. Sử dụng tối đa lượng tài nguyên có thể sử dụng trên một node.*

*2. Migrate và thực hiện replacement các task giữa các node để đạt được luật 1. (Dồn task từ node dư nhiều tài nguyên sang node còn dư đủ tài nguyên để chạy task)*

*3. Thực hiện di dời task trên một node khi lượng tài nguyên của node sắp cạn kiệt sang node khác sao cho node gốc và node mới thỏa mãn luật 1. (ưu tiên di chuyển đến node còn tài nguyên vừa đủ để thực hiện task)*

*4. Đảm bảo các node không bị quá tải*

**4.1.2 Áp dụng thuật toán Y và dựa trên tập luật cho bộ Engine scheduling**

- Sử dụng kết hợp luật và thuật toán Y để ứng dụng vào bộ engine scheduling

**4.2 Prediction scheduling**

**4.2.1 Sử dụng phương pháp Linear Regestion để dự đoán mức tiêu thụ tài nguyên của 1 task/ sensor**

- Dự đoán mức tiêu thụ tài nguyên của 1 task/ sensor để có cơ sở áp dụng vào việc assign node cho platform instance. (Giải quyết vấn đề đặt ra trong phần 3)

**4.2.2 Sử dụng phương pháp X để phát hiện chu kỳ tiêu thụ tài nguyên của 1 task/ sensor**

- Phát hiện chu kì để từ đó phán đoán ra quyết định schedule.