

LỜI CẢM ƠN

Đầu tiên, tôi xin được gửi lời cảm ơn sâu sắc nhất tới Thầy giáo - TS Trần Hoàng Hải - Giảng viên, Viện Công nghệ thông tin và Truyền thông, Trường Đại học Bách Khoa Hà Nội đã hướng dẫn và cho tôi những lời khuyên trong quá trình thực hiện luận văn này.

Tiếp theo, tôi xin chân thành cảm ơn các thầy cô trong Viện Công nghệ thông tin và truyền thông, Viện đào tạo sau đại học, Trường Đại học Bách Khoa Hà Nội đã tạo điều kiện cho tôi trong suốt quá trình học tập và nghiên cứu tại trường.

Cuối cùng, tôi xin bày tỏ lòng cảm ơn tới những người thân trong gia đình, bạn bè đã động viên và giúp đỡ để tôi hoàn thành bản luận văn này.

Hà Nội, ngày... tháng... năm 2020

Tác giả

Hồ Đình Tân

MỤC LỤC

LỜI CẢM ƠN	1
MỤC LỤC	2
DANH MỤC KÝ HIỆU, CÁC CHỮ VIẾT TẮT	6
DANH MỤC CÁC BẢNG BIỂU	6
DANH MỤC HÌNH VẼ	6
MỞ ĐẦU	7
Chương 1: LÝ THUYẾT	11
1.1 Hệ thống IoT là gì ?	11
1.1.1 Định nghĩa về IoT	11
1.1.2 Tầm quan trọng của IoT	12
1.1.3 Ứng dụng của IoT trong các ngành công nghiệp	13
1.3.3.1 Ngành chế tạo	14
1.3.3.2 Ngành ô tô	14
1.3.3.3 Giao thông vận tải	15
1.3.3.4 Ngành bán lẻ	15
1.3.3.5 Chăm sóc sức khỏe	15
1.2. Microservice là gì ?	16
1.2.1 Kiến trúc một khối (Monolithic Architecture)	16
1.2.1.1 Định nghĩa kiến trúc một khối	16
1.2.1.2 Ưu điểm:	18
1.2.1.3 Địa ngục kiến trúc một khối	18
1.2.2 Microservice - đơn giản hóa sự phức tạp	21

1.2.3 Các thuộc tính của mô hình microservice	23
1.3 Networking là gì ?	26
1.3.1 Wired & Short Range wireless networks Mạng dây và Không dây tầm ngắn	27
1.3.2 M2M – 2G, 3G, 4G & 5G networks	27
1.3.3 LPWAN – Low Power Wide Area Networks	29
1.3.4 SigFox & LoRaWAN	30
1.3.5 NB-IOT (Narrow Band IOT)	33
1.4 Tại sao tôi lại đề cập đến vấn đề này trong khi nhiều người đã làm trước đó, và có gì khác biệt với các công nghệ trước đó ? Tôi định làm cái gì ?	34
Chương 2: THINGSBOARD - IoT Platform	35
2.1 Thingsboard là gì ?	35
2.1.1 What is ThingsBoard?	35
2.1.2 Đặc trưng	35
2.2 Các công nghệ thingsboard sử dụng ?	36
2.1.1 Tính năng của Phiên bản Cộng đồng	36
2.1.2 Tính năng của Phiên bản Chuyên nghiệp	37
2.1.3 Bảo vệ	39
2.1.4 Mẫu	39
2.1.5 API	39
2.1.6 Cổng IoT	39
2.1.7 Video hướng dẫn	40
2.1.8 Các tính năng của Trendz Analytics	40
2.3. Thingsboard với microservices	40

2.3.1. ThingsBoard Monolithic architecture	40
2.3.2 ThingsBoard Microservices architecture	41
2.3.2.1 Dịch vụ vi mô vận tải	41
2.3.2.2 Web UI Microservices	43
2.3.2.3 JavaScript Executor Microservices	44
2.3.2.4 ThingsBoard Node	45
2.3.2.5 Third-party	46
2.3.2.5.1 Kafka **	46
2.3.2.5.1 Redis **	46
2.3.2.5.2 Zookeeper **	46
2.3.2.5.3 HAProxy (hoặc LoadBalancer khác) **	47
2.3.2.5.4 Cơ sở dữ liệu **	47
Chương 3: THIẾT KẾ GIẢI PHÁP GIẢ LẬP ỨNG DỤNG QUẢN LÝ THƯ VIỆN TẠ QUANG BỬU	47
3.1 Mục đích	48
3.1.1 Giám sát, quản lý, cảnh báo trong thư viện Tạ Quang Bửu	48
3.2 Thiết kế tổng thể	48
3.2.1 sensors nhiệt độ, báo cháy, báo khói	48
3.2.2 cameras	48
3.2.3 còi	48
3.2.4 điều hòa	48
3.2.5 màn hình giám sát	48
3.3 Thiết kế chi tiết	48
3.3.1 Giả lập hệ thống cảm biến, cảnh báo, camera, máy tính,	48
3.3.2 Thiết kế hệ thống thiết bị tương ứng thực tế	48

3.3.3 Thiết kế Widgets giao diện ứng dụng	48
3.4 Kết nối dữ liệu đến Thingsboard	48
3.4.1 MQTT message	48
3.4.2 API	48
3.5 Lưu trữ	49
3.5.1 MongoDB	49
3.6 Phân phối dữ liệu.	49
3.6.1 Phân phối tải cho server	49
Chương 4: KẾT LUẬN	49

DANH MỤC KÝ HIỆU, CÁC CHỮ VIẾT TẮT

Từ viết tắt	Nghĩa tiếng Anh	Nghĩa tiếng Việt
IoT		
AI		

DANH MỤC CÁC BẢNG BIỂU

DANH MỤC HÌNH VẼ

MỞ ĐẦU

Cuộc cách mạng công nghiệp (CMCN) lần thứ tư đã và đang đến. Đây là cuộc cách mạng chưa từng có trong lịch sử nhân loại, nó sẽ diễn biến rất nhanh, là sự kết hợp của công nghệ trong các lĩnh vực vật lý, số hóa và sinh học, tạo ra những khả năng hoàn toàn mới và có tác động sâu sắc đối với các hệ thống chính trị, xã hội, kinh tế của thế giới trong đó dựa trên các lĩnh vực cốt lõi như Trí tuệ nhân tạo (AI), Kết nối vạn vật (IoT) và Dữ liệu lớn (Big Data). Các lĩnh vực này liên kết chặt chẽ với nhau để tạo ra xu hướng công nghệ 4.0 như hiện nay.

Trong đó IoT là lĩnh vực có phạm vi ứng dụng lớn lao và cần sự có mặt thiết gần như tuyệt đối của Big Data đồng thời có thể sử dụng hiệu quả sức mạnh của [AI. Do](#) đó IoT là vấn đề hiện nay đang được rất nhiều người trên thế giới quan tâm đến công nghệ thông tin nhắc đến. Không chỉ là vì tính ứng dụng cao mà nó còn được coi là công nghệ tiêu biểu của công nghiệp 4.0.

IoT đang là thứ mà các tập đoàn công nghệ lớn đặc biệt quan tâm và giờ họ vẫn đang đầu tư hàng tỷ đô la vào đây. Có thể ví IoT như là một nông trại cực kỳ rộng lớn và trù phú, nơi họ có thể gieo trồng và thu lại lợi nhuận gần như lớn gấp chục lần những gì mà họ bỏ ra. Những người đam mê công nghệ, những chuyên viên máy tính và kỹ sư lập trình cũng là các đối tượng không thể bỏ qua IoT. Đơn giản là vì cái họ được tiếp cận, được thực hành đều là những ứng dụng, nền tảng của tương lai và đó sẽ là trải nghiệm tuyệt vời nhất đối với họ.

Tại Việt Nam IoT đã được ứng dụng từ lâu dưới các hình thức tự động hóa như hệ thống điều khiển đèn giao thông, hệ thống tưới tiêu tự động,... Tuy nhiên chỉ đến những năm gần đây thì khái niệm IoT tại Việt Nam mới được nhắc đến nhiều thông qua các hội thảo, hội nghị về xu hướng công nghệ của Cisco, Intel và một số công ty trong nước như Viettel, Mobifone, Vinaphone, ... Trước đó, IBM có chiến dịch

“Hành tinh thông minh hơn” và nhấn mạnh vào các thành phố thông minh trong đó Đà Nẵng được chọn thực hiện thí điểm này từ năm 2012-2013.

Ở thời điểm hiện tại Việt Nam đang có rất nhiều công ty tập trung phát triển giải pháp và sản phẩm công nghệ thông minh với nền tảng IoT. Có thể kể đến những cái tên quen thuộc và được thị trường dần đón nhận trong thời gian vừa qua như: Lumi, BKAV, SmartHome,... Một điểm chung dễ nhận thấy ở các nhà cung cấp này là họ tập trung vào thiết bị nhà ở thông minh (SmartHome) hướng tới đối tượng khách hàng là những người sẵn sàng bỏ chi phí để tiện dụng hóa các hoạt động trong gia đình. Các sản phẩm này được đầu tư khá bài bản về mặt hình thức nhằm giúp cho căn nhà trở nên sang trọng hơn.

Một nghiên cứu gần đây của Cisco mang tên “Ready, Steady, Unsure” (Sẵn sàng, Ổn định, Không chắc chắn) cho biết: Hơn một nửa công ty ở Việt Nam được hỏi đã xếp IoT là một trong ba công nghệ hàng đầu sẽ tác động đến tương lai kỹ thuật số của doanh nghiệp họ. Nghiên cứu này cũng chỉ ra rằng 36% công ty tham gia trả lời đã bắt đầu sử dụng các giải pháp IoT. Đây là tỷ lệ cao nhất ở Đông Nam Á, ngang với đất nước Singapore.

Tuy nhiên, ở chiều ngược lại các doanh nghiệp còn ngại ứng dụng công nghệ này bởi các lý do như: cơ sở hạ tầng yếu kém (31%), đội ngũ nhân viên chưa đủ nhân lực (20%), thiếu ngân sách (44%) và chưa chắc chắn vào lợi ích cũng như tác động của IoT đối với doanh nghiệp (48%). Lợi ích của IoT thì đã rõ nhưng điều khó khăn là vẫn còn một số bộ phận doanh nghiệp, tổ chức, cá nhân, chính quyền vẫn chưa sẵn sàng áp dụng IoT tại Việt Nam. Các hội thảo, tham luận về đô thị thông minh, giao thông thông minh ngày càng nhiều nhưng sự thông minh đó vẫn chưa áp dụng được nhiều vào cuộc sống.

Trong tương lai IoT chắc chắn sẽ là một trong những mũi nhọn cho phát triển đất nước, Vì thế việc nghiên cứu và chuẩn bị cho làn sóng đó là cần thiết. Trong luận

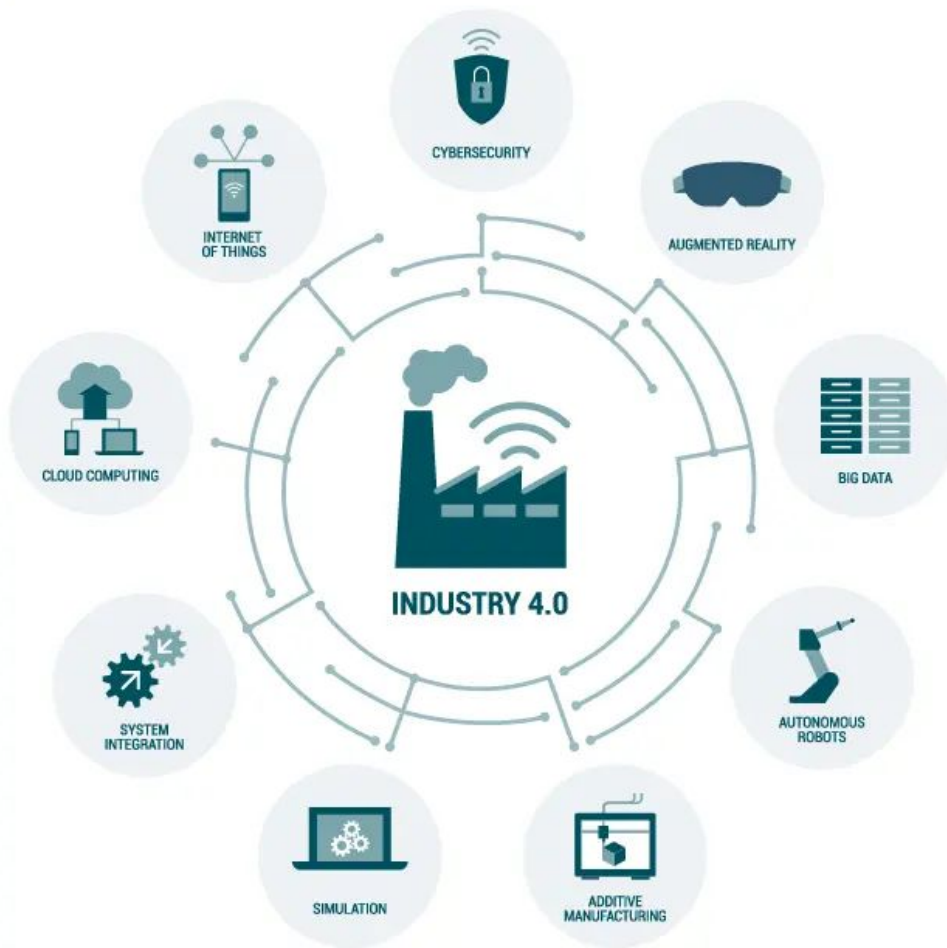
văn này Em mong đóng góp một số tìm hiểu về cấu trúc và vận hành demo đơn giản một hệ thống IoT sử dụng Microservices- cũng là một công nghệ được ưa chuộng sử dụng hiện nay để có cái nhìn chi tiết hơn về mặt ứng dụng của IoT.

Hình 1.1 Cách mạng công nghiệp 4.0

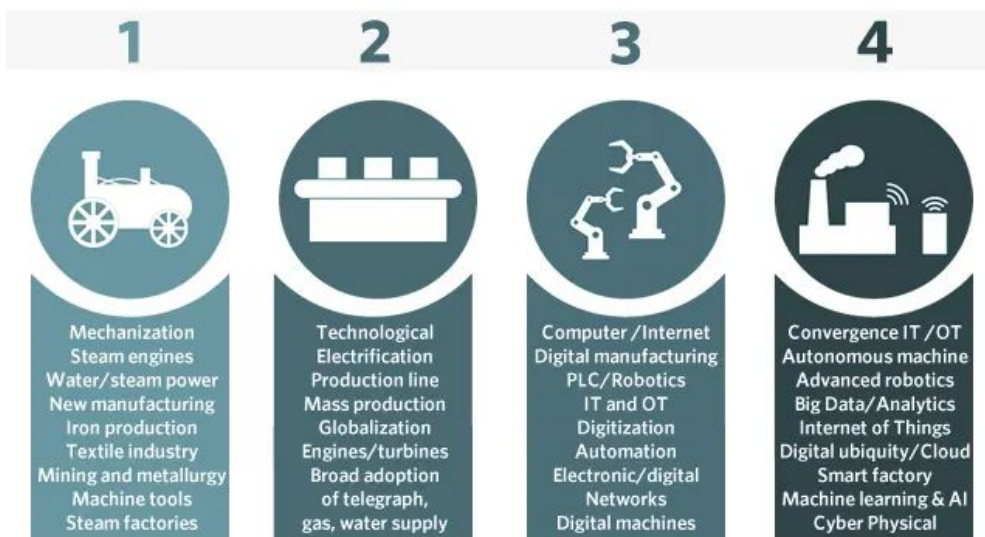
INDUSTRY 4.0 - the digital transformation



3rd platform, innovation accelerators, OT and manufacturing meet in transformation



FROM INDUSTRY 4.0 TO FOURTH INDUSTRIAL REVOLUTION



Chương 1: LÝ THUYẾT

1.1 Hệ thống IoT là gì ?

IoT – Internet of Things là thời kỳ mà các thiết bị xung quanh chúng ta đều trở nên thông minh và được kết nối tới nhưng dịch vụ trên mạng. Những thiết bị có sử dụng IoT thường được lập trình và có khả năng tự động làm việc, giúp cho cuộc sống của chúng ta thông minh hơn, tiết kiệm thời gian và chi phí trong cuộc sống hàng ngày.

1.1.1 Định nghĩa về IoT

Ngày nay thuật ngữ này được đưa ra với vô vàn khái niệm và chúng tôi xin đưa ra một khái niệm để các bạn có thể dễ hiểu nhất.

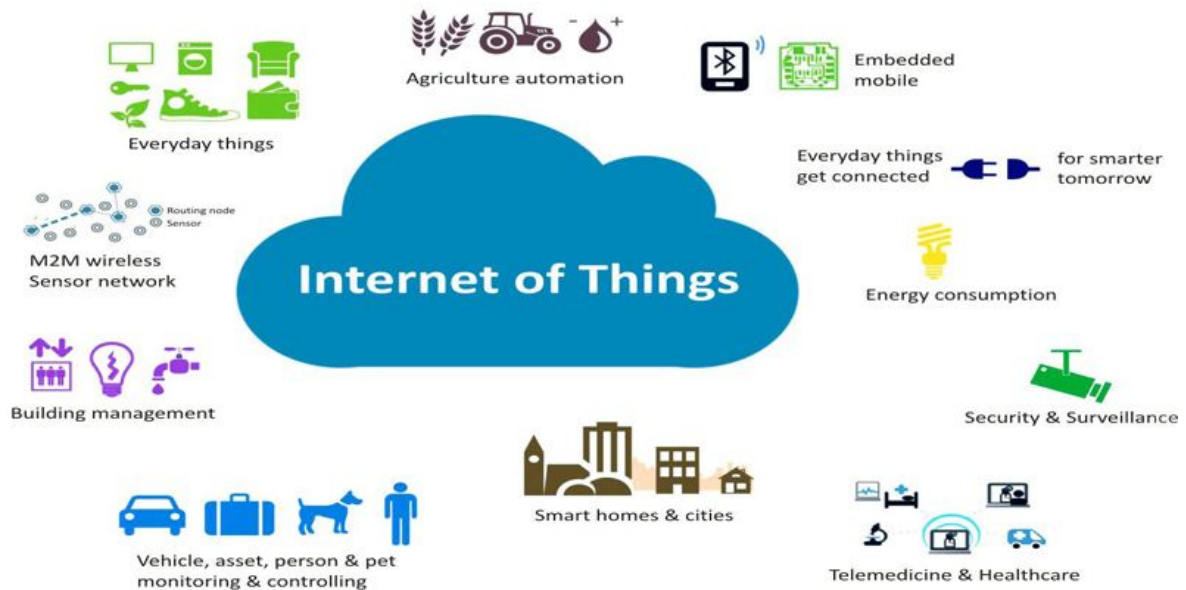
Internet of Things hay IoT đề cập đến hàng tỷ thiết bị vật lý trên khắp thế giới hiện được kết nối với internet, thu thập và chia sẻ dữ liệu. Nhờ bộ xử lý bên trong cùng mạng không dây, bạn có thể biến mọi thứ trở nên chủ động và thông minh hơn.

Ta có thể bắt gặp IoT từ hệ thống cửa tự động cho tới máy bay tới xe tự lái đã trở thành một phần phổ biến của IoT. Điều này bổ sung một mức độ thông minh kỹ thuật số cho các thiết bị thụ động, cho phép chúng giao tiếp dữ liệu thời gian thực mà không cần con người tham gia, hợp nhất hiệu quả thế giới kỹ thuật số và vật lý.

Internet Of Things: là một kịch bản của thế giới, khi mà mỗi đồ vật được cung cấp một định danh của riêng mình, và tất cả có khả năng truyền tải, trao đổi thông tin, dữ liệu qua một mạng duy nhất mà không cần đến sự tương tác trực tiếp giữa người với người, hay người với máy tính. IoT đã phát triển sự hội tụ của công nghệ không dây, công nghệ cơ điện tử và Internet. Nói đơn giản là một tập hợp các thiết bị có khả năng kết nối với nhau, với Internet và với thế giới bên ngoài để thực hiện một công việc nào đó.

1.1.2 Tầm quan trọng của IoT

Khi bất cứ vật gì đó được kết nối với internet, điều đó có nghĩa là nó có thể gửi thông tin hoặc nhận thông tin, hoặc cả hai. Với IoT khả năng gửi hoặc nhận thông tin này làm cho mọi thứ trở nên thông minh, và thông minh luôn là điều hướng đến.



Sử dụng lại điện thoại thông minh (điện thoại thông minh) làm ví dụ. Ngay bây giờ bạn có thể nghe bất kỳ bài hát nào trên thế giới, không phải vì điện thoại của bạn thực sự có mọi bài hát trên thế giới được lưu trữ trong nó. Nó có nghĩa là vì mọi bài hát trên thế giới đều được lưu trữ ở một nơi khác, nhưng điện thoại của bạn có thể gửi thông tin (yêu cầu bài hát đó) và sau đó nhận thông tin (phát trực tuyến bài hát đó trên điện thoại của bạn).

Để trở nên thông minh, một thứ không cần phải có siêu lưu trữ hoặc siêu máy tính bên trong nó. Tất cả những gì phải làm là kết nối với siêu lưu trữ hoặc với một siêu máy tính

Trong Internet of Things, tất cả những thứ đang được kết nối với internet có thể được chia thành ba loại:

Loại thu thập thông tin và sau đó gửi nó.

Ví dụ: các thiết bị mang tính cảm biến, có thể là cảm biến nhiệt độ, cảm biến chuyển động, cảm biến độ ẩm, ánh sáng,... Những cảm biến này cùng với một kết nối, cho phép chúng ta tự động thu thập thông tin từ môi trường. Do đó, cho phép chúng tôi đưa ra quyết định thông minh hơn.

Loại nhận được thông tin và sau đó hành động.

Ví dụ: máy in của bạn nhận được một tài liệu và in nó. Xe của bạn nhận được tín hiệu từ chìa khóa xe và cửa mở.








Thực hiện cả hai.

Lấy một ví dụ hiện đang được sử dụng rộng rãi trong ngành nông nghiệp. Các cảm biến có thể thu thập thông tin về độ ẩm của đất để cho nông dân biết cần tưới bao nhiêu cho cây trồng, nhưng bạn không thực sự cần người nông dân. Thay vào đó, hệ thống tưới có thể tự động bật khi cần thiết, dựa trên độ ẩm của đất.

Thêm vào đó, các nhà nông nghiệp đã phát minh thêm một bước tiến nữa. Nếu hệ thống thủy lợi nhận được thông tin về thời tiết từ kết nối internet của nó, thì nó cũng có thể biết khi nào trời sẽ mưa và quyết định không tưới nước cho các loại cây trồng ngày hôm nay vì tận dụng được nguồn nước mưa.

1.1.3 Ứng dụng của IoT trong các ngành công nghiệp

Cùng tham khảo một số lĩnh vực công nghiệp đã và đang áp dụng IoT tích hợp vào bên trong như thế nào.

Transport & Logistics  Fleet management, Goods tracking	Utilities  Smart metering, Smart grid management	Smart cities  Parking sensors, Waste management, etc.	Smart building  Smoke detector, Home automation
Consumers  Wearables Kids/senior tracker	Industrial  Process monitoring & control, Maintenance monitoring	Environment  Food monitoring/alerts, Environmental monitoring	Agriculture  Climate/agriculture monitoring, Livestock tracking

1.3.3.1 Ngành chế tạo

Các nhà sản xuất có thể đạt được lợi thế cạnh tranh bằng cách sử dụng giám sát dây chuyền sản xuất để cho phép bảo trì chủ động trên thiết bị khi cảm biến phát hiện ra lỗi sắp xảy ra.

Các cảm biến thực sự có thể đo lường khi sản lượng sản xuất bị tổn hại. Với sự trợ giúp của cảnh báo cảm biến, các nhà sản xuất có thể nhanh chóng kiểm tra độ chính xác của thiết bị hoặc loại bỏ nó khỏi sản xuất cho đến khi nó được sửa chữa. Điều này cho phép các công ty giảm chi phí hoạt động, có thời gian hoạt động tốt hơn và cải thiện quản lý hiệu suất tài sản.

1.3.3.2 Ngành ô tô

Ngành công nghiệp ô tô đã nhận ra những lợi thế đáng kể từ việc sử dụng các ứng dụng IoT. Ngoài những lợi ích của việc áp dụng IoT vào dây chuyền sản xuất, các cảm biến có thể phát hiện lỗi thiết bị sắp xảy ra trên các phương tiện đã đi trên đường và có thể cảnh báo cho người lái xe một cách chi tiết.

Nhờ thông tin tổng hợp được thu thập bởi các ứng dụng dựa trên IoT, các nhà sản xuất và nhà cung cấp ô tô có thể tìm hiểu thêm về cách giữ cho xe chạy và thông báo cho chủ xe về các thông tin phía trước.

1.3.3.3 Giao thông vận tải

Các đội xe ô tô, xe tải và tàu chở hàng tồn kho có thể được định tuyến lại dựa trên điều kiện thời tiết, tính sẵn có của xe hoặc tính khả dụng của tài xế, nhờ dữ liệu cảm biến IoT. Bản thân hàng tồn kho cũng có thể được trang bị các cảm biến để theo dõi và kiểm soát nhiệt độ.

Các ngành công nghiệp thực phẩm và đồ uống, hoa và dược phẩm thường là những mặt hàng tồn kho nhạy cảm với nhiệt độ sẽ được hưởng lợi rất nhiều từ các ứng dụng giám sát IoT gửi thông báo khi nhiệt độ tăng hoặc giảm có thể ảnh hưởng đến sản phẩm.

1.3.3.4 Ngành bán lẻ

Các ứng dụng IoT cho phép các công ty bán lẻ quản lý hàng tồn kho, cải thiện trải nghiệm của khách hàng, tối ưu hóa chuỗi cung ứng và giảm chi phí hoạt động.

Ví dụ: kệ thông minh được trang bị cảm biến trọng lượng có thể thu thập thông tin dựa trên RFID và gửi dữ liệu tới nền tảng IoT để tự động theo dõi hàng tồn kho và kích hoạt cảnh báo nếu các mặt hàng sắp hết.

1.3.3.5 Chăm sóc sức khỏe

IoT cung cấp nhiều lợi ích cho ngành chăm sóc sức khỏe. Bác sĩ, y tá thường cần biết chính xác vị trí của các tài sản hỗ trợ bệnh nhân như xe lăn. Khi xe lăn của bệnh viện được trang bị cảm biến IoT, chúng có thể được theo dõi từ ứng dụng giám sát tài sản IoT để bất kỳ ai đang tìm kiếm đều có thể nhanh chóng tìm thấy chiếc xe lăn có sẵn gần nhất.

Nhiều tài sản của bệnh viện có thể được theo dõi theo cách này để đảm bảo sử dụng hợp lý cũng như kế toán tài chính cho các tài sản vật chất trong mỗi khoa.

1.2. Microservice là gì ?

Nền tảng của kiến trúc microservices là xây dựng một ứng dụng mà ứng dụng này là tổng hợp của nhiều services nhỏ và độc lập có thể chạy riêng biệt, phát triển và triển khai độc lập.

Microservices hiện được quan tâm trong giới phần mềm, công nghệ với nhiều bài viết, blog, thảo luận, truyền thông, hội thảo. Kỳ vọng về khả năng của Microservice đang lên đỉnh giống như một xu hướng thời trang đang lan rộng. Ngược lại, một số người cho rằng, microservices không có gì mới lạ, chẳng qua nó là SOA (kiến trúc hướng dịch vụ) được đánh bóng, đổi tên mà thôi.

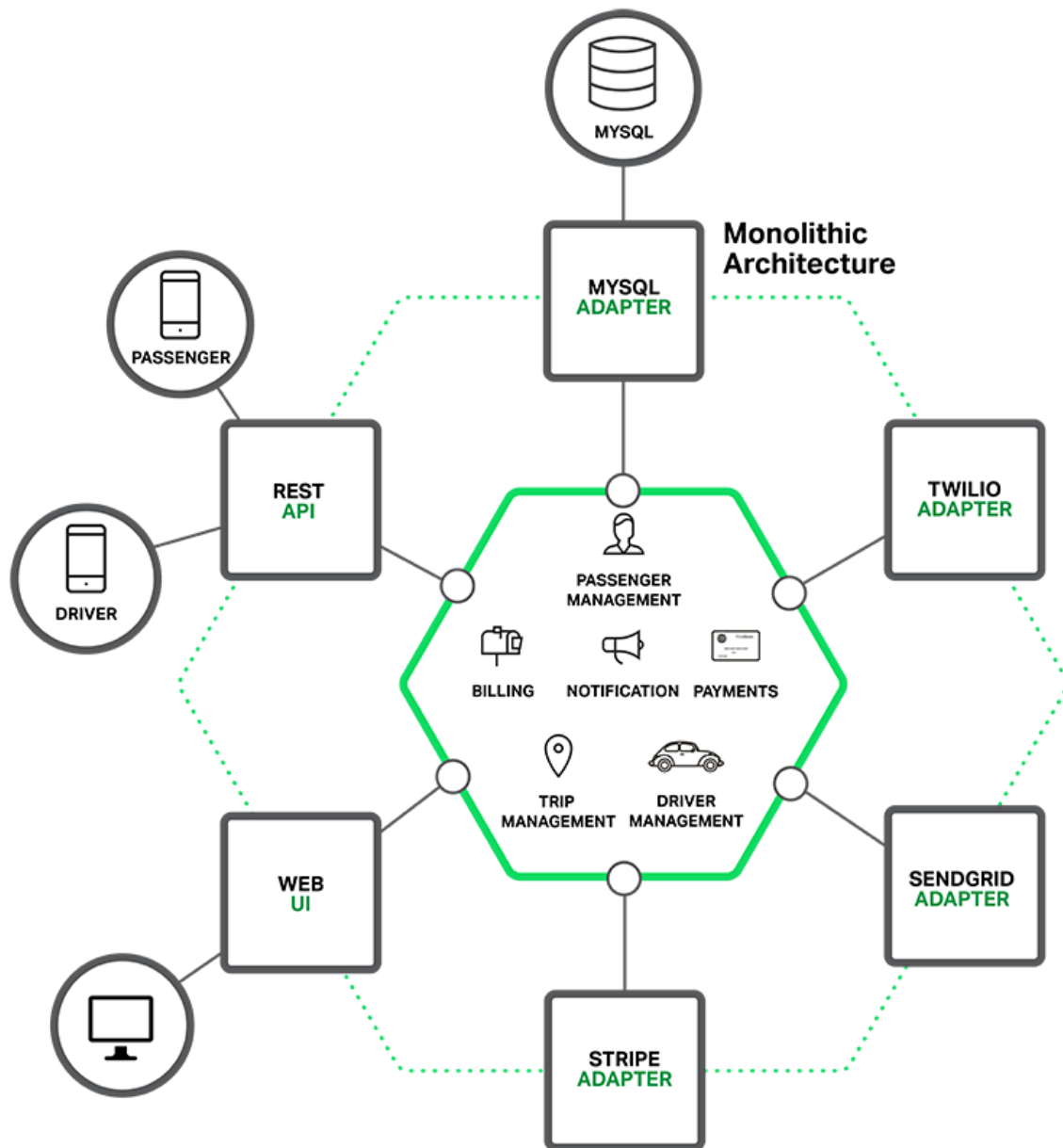
Mặc cho kỳ vọng cao, hay đánh giá bảo thủ, kiến trúc microservices vẫn đem lại lợi ích khi nó giúp phương pháp agile thực sự hiệu quả và xây dựng được giải pháp phần mềm doanh nghiệp rất phức tạp.

Bài viết này sẽ giải thích tại sao công ty, tổ chức của bạn nên cân nhắc áp dụng kiến trúc microservices

Để hiểu rõ hơn về microservice thì chúng ta cần so sánh nó kiến trúc đối lập với nó là kiến trúc một khối (Monolithic Architecture).

1.2.1 Kiến trúc một khối (Monolithic Architecture)

1.2.1.1 Định nghĩa kiến trúc một khối



Một dịch vụ gọi taxi qua di động, với business logic được thể hiện bởi các khối dịch vụ, đối tượng cho từng vùng nghiệp vụ (domain objects) và các sự kiện (events: khách đặt xe, khách hủy xe, xe nhận khách...) Xung quanh lõi là bộ chuyển đổi (adapter) ví dụ như kết nối vào cơ sở dữ liệu, gửi nhận thông điệp (messaging), web service hoặc giao diện web front end.

Trong nhiều trường hợp, người ta có thể xây dựng các services độc lập nhưng chúng lại được triển khai chung. Mặc dù có cấu trúc module hóa hợp lý, nhưng ứng dụng kiểu này sẽ đóng gói và cài đặt thành một khối (monolithic). Mã chạy cụ thể tùy thuộc vào ngôn ngữ lập trình hay thư viện framework.

1.2.1.2 Ưu điểm:

- Dễ phát triển vì các stack công nghệ thống nhất ở tất cả các layer.
- Dễ test do toàn bộ project được đóng gói trong một package nên dễ dàng chạy test integration và test end-to-end.
- Deploy đơn giản và nhanh chóng nếu bạn chỉ có một package để bận tâm.
- Dễ scale vì chúng ta có thể có nhiều instance cho load balancer.
- Yêu cầu team size nhỏ cho việc maintain app.
- Team member có thể chia sẻ ít nhiều về skill.
- Tech stack đơn giản và đa số là dễ học.
- Phát triển ban đầu nhanh hơn do đó có thể đem sale hoặc marketing nhanh hơn.
- Yêu cầu cơ sở hạ tầng đơn giản. Thậm chí một container đơn giản cũng đủ để chạy ứng dụng.

1.2.1.3 Địa ngục kiến trúc một khối

Đáng tiếc rằng, cách tiếp cận kiến trúc đơn nhất tuy dễ dàng nhưng bắt đầu bộc lộ nhiều khiếm khuyết. Ứng dụng thành công - số lượng người dùng tăng - yêu cầu tính năng mới tăng - dữ liệu tăng - logic phức tạp hơn - giao tiếp với hệ thống khác tăng kết quả một ứng dụng khủng. Sau mỗi kỳ phát triển (sprint), đội phát triển bổ xung vài tính năng mới, thêm code, thêm bảng, thêm logic... Chỉ sau vài năm, ứng dụng đơn giản sẽ kèn càng như quái vật. Tôi có trao đổi với một lập trình viên, người từng viết công cụ phân tích sự phụ thuộc giữa hàng nghìn gói thư viện JAR trong

ứng dụng hàng triệu dòng code. Chắc chắn một số lượng lớn man month và tiền tấn để tạo ra quái vật khủng đến vậy.

Ứng dụng một khối mà phình to sẽ rắc rối như một gia đình nhiều thế hệ đông con cái ở trong cùng một nhà. Nhà to đến mấy rồi cũng sẽ gặp vấn đề. Mọi nỗ lực tối ưu, phương pháp làm việc agile (mềm dẻo) đều không còn hiệu quả. Một thư viện được tham chiếu nhiều chỗ khi nâng cấp sẽ phải kiểm tra ở tất cả những điểm, nghiệp vụ mà nó được gọi. Ứng dụng đơn nhất dùng một ngôn ngữ lập trình duy nhất. Đội lập trình sẽ quen với sự thuận tiện, dễ dàng khi chỉ cần nắm sâu một ngôn ngữ, một công cụ có thể giải quyết hầu hết vấn đề. Họ dần lệ thuộc vào ngôn ngữ đó và trở nên thiên vị, ngại cởi mở, tích hợp với những công nghệ khác của ngôn ngữ khác. Thậm chí khi framework hay ngôn ngữ có nhược điểm cố hữu, tính đơn nhất một khối của ứng dụng sẽ bó buộc lập trình viên thử nghiệm đưa vào thay đổi đột phá ngoại lai.

Ứng dụng một khối có hơn 2 triệu dòng mã trên framework XYZ, liệu đội bạn có đủ dũng cảm, nguồn lực để viết lại toàn bộ trên framework ABC mới hơn, tốt hơn. Lập trình giỏi, sáng tạo cũng không muốn làm trong kiểu dự án mặc kệ này. Tình trạng giữ thì khổ, xây thì khó, khiến dịch vụ - sản phẩm của bạn ì ạch kém cạnh tranh so với các dịch vụ mới nổi uyển chuyển, linh hoạt.

Trong ứng dụng một khối, sự chặt chẽ là ưu điểm tự nhiên xuất phát từ kiến trúc, nhưng nó tiềm ẩn nguy cơ ràng buộc cứng nhắc đóng bê tông (tight coupling). Chi phí, thời gian, nỗ lực phát triển - sửa lỗi - kiểm thử một chức năng sẽ tăng tỷ lệ bậc 2 theo độ lớn của ứng dụng. Nói cách khác đi, mã nguồn khó đọc, khó bảo trì tỷ lệ bậc 2 theo số ràng buộc, tham chiếu được tạo ra hết sức dễ dãi khi phát triển.

Ở trên tôi có nói đến khả năng biên dịch nóng - khởi động lại (hot reload) khi code đổi, hay cân bằng tải bằng cách thêm nhiều ứng dụng web giống sau sau bộ cân bằng tải, nhưng khi ứng dụng to khủng, việc biên dịch nóng kéo dài hơn, khởi động

lại sẽ chậm đi, copy phiên bản mới ra các server sẽ lâu hơn. Có những ứng dụng thời gian khởi động kéo dài từ 12-40 phút. Việc lập trình và gỡ rối tệ như để thay một con ốc trên đoàn tàu hỏa đang chạy, mà chúng ta phải dừng cả đoàn tàu rồi khởi động lại.

Gần đây, bạn nghe nói nhiều hơn về triển khai đều đặn (continuous deployment). Những ứng dụng SaaS (Software application as Service) tiên tiến, cần phải cập nhật vài lần trong một ngày. Quá khó để triển khai lại cả một ứng dụng cực lớn chỉ vì một số nâng cấp nhỏ. Hoạt động bị ngưng trệ, kiểm thử lại sau triển khai sẽ lâu công hơn. Kết quả là triển khai đều đặn khó áp dụng với ứng dụng một khối.

Khả năng mở rộng chịu tải ứng dụng một khối sẽ khó khi các thành phần khác nhau tranh chấp, dị biệt nhu cầu dùng tài nguyên hệ thống. Ví dụ module xử lý ảnh cần triển khai trên Amazon EC2 tối ưu CPU, sẽ khó cho module lưu bộ nhớ tạm (cache) cần rất nhiều bộ nhớ đang ra phải triển khai trên EC2 tối ưu bộ nhớ.

Tóm lại, bám vào kiến trúc một khối, một vé xuống địa ngục là cao hơn lên thiên đường.

Monolith có xu hướng phù hợp với những dự án có quy mô nhỏ. Với việc áp dụng mô hình monolith, những lợi ích đem lại có thể kể đến là:

- Quá trình development đơn giản và trực tiếp, centralized management và những bước phát triển cơ bản thì sẽ không được lặp lại.
- Effort dành cho việc development được giảm thiểu: tất cả mọi quá trình development đều nằm trên 1 project. Development flow đơn giản chỉ là submit changes, review, merge code và continue.

Tuy nhiên hạn chế mà mô hình này đem lại cũng khá lớn :

- Khó khăn trong việc bảo trì: vấn đề về coupling code, các khối code dính chặt lại với nhau, vấn đề cho member mới sẽ khó để biết nên bắt đầu từ đâu trong 1 khối lớn
- Quá trình development sẽ mất đi tính linh hoạt: thời gian để build feature sẽ bị dài lên, bị block lẫn nhau. Bất kì một sự thay đổi dù nhỏ nào cũng cần build lại toàn bộ dự án => tốn khá nhiều thời gian
- Tính ổn định không cao. Bất kì một lỗi nào có thể khiến toàn bộ application bị crash.
- Tính scalability khó được đáp ứng trong trường hợp phải đáp ứng một lượng truy cập lớn từ phía yêu cầu của business

1.2.2 Microservice - đơn giản hóa sự phức tạp

Ngoài mô hình monolithic kể trên, hiện nay có 1 architecture khác đang nhận được nhiều sự quan tâm, đó là microservice. Microservice đề cập đến quá trình phát triển độc lập, tương đối nhỏ theo hướng chia hệ thống ra thành các services. Mỗi service này đều có một logic riêng, một trách nhiệm riêng và có thể được deploy riêng biệt. Khái niệm microservice đồng thời đề cập đến xu hướng tách biệt architecture ra thành các loose coupling service, tức là các service này sẽ có một mối liên hệ lỏng lẻo với nhau và mỗi service sẽ được nằm trong 1 context nhất định.

So sánh với microservice và SOA (service-oriented architecture), những điểm khác biệt của mô hình microservice là componentization (thành phần hóa), loose coupling (khớp nối lỏng lẻo), autonomy (tính tự quản lí) và decentralization (phân cấp), được phản ánh cụ thể qua những khía cạnh sau:

- tập hợp một nhóm nhỏ các service: mức độ chi tiết của một service là nhỏ và mỗi service này sẽ chịu một trách nhiệm cụ thể (single responsibility) và chỉ tập trung vào nhiệm vụ đó. Ví dụ: storage service sẽ chịu riêng trách nhiệm về lưu trữ

- Việc phát triển và mở rộng một service là hoàn toàn độc lập. Điều này mang lại tính linh hoạt cho hệ thống . Quá trình deliver feature, release version sẽ dễ dàng và nhanh chóng. Hơn nữa sẽ không còn tình trạng bị block như ở mô hình monolith
- Giảm tải được các mối quan ngại về công nghệ sử dụng. Chọn một công nghệ phù hợp với vấn đề của doanh nghiệp có thể được giải quyết dễ dàng. Các service giao tiếp với nhau thông qua API, do vậy mỗi service có thể dùng một ngôn ngữ riêng biệt. Service A dùng Java, Service B dùng Javascript, it's ok !!!!
- Đối với team, microservice đem lại tính độc lập và tự quản lí cho team. Một team sẽ có trách nhiệm toàn bộ với life-cycle của một hay nhiều service. Họ làm việc trong việc context biệt lập, có thể tự quản lí các quyết định của mình.

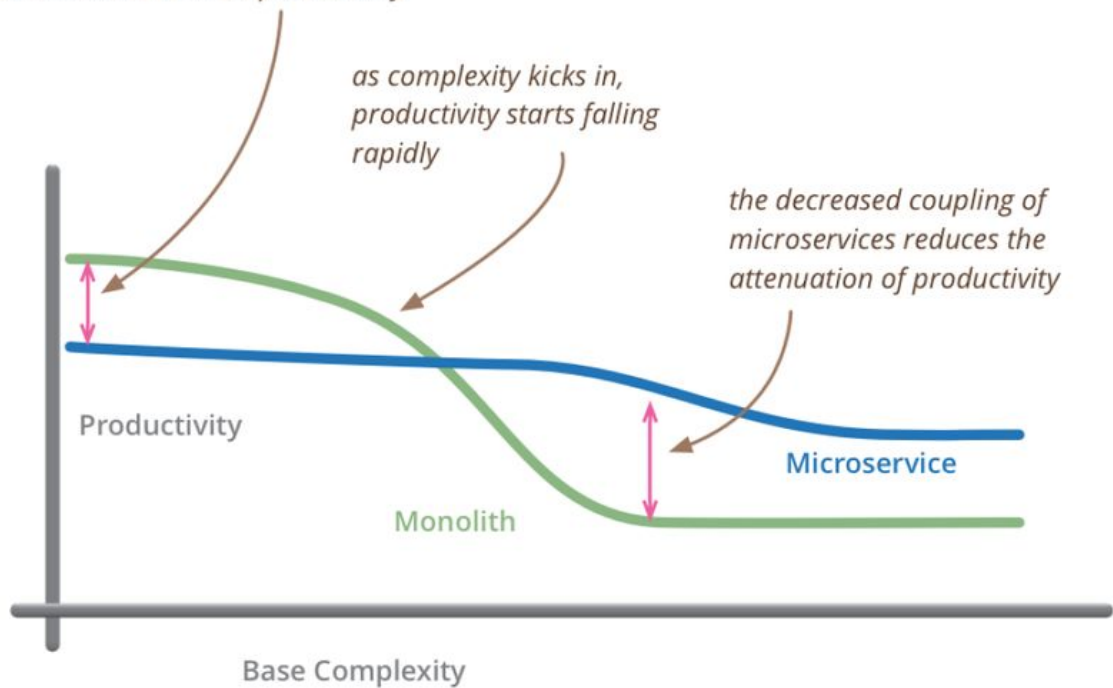
Chúng ta có thể thấy rõ toàn bộ ý tưởng của mô hình microservice rất giống cách mà chúng ta chia nhỏ thông tin và kiến thức. Bằng việc tách rời, chia nhỏ và quản lí chúng ta có thể giảm tải sự phức tạp của hệ thống, làm cho việc quản lí trở nên nhanh chóng và dễ dàng, phản ánh sự thay đổi chính xác.

Vậy tại sao chúng ta nên dùng microservice ?

Ở thế kỷ trước, một số lightweight development methods như eXtreme Programming (XP) hay Scrum nổi lên; Đến năm 2001, tuyên ngôn Agile ra đời và một số phương pháp quản lí mới như Lean hay Kanban. Nếu những phương pháp quản lí trên được coi là giải pháp cho việc quản lí tiến độ phát triển phần mềm và việc thực hiện sớm nhất có thể khi có sự thay đổi thì microservice architecture là hướng tiếp cận được nói đến trong công nghệ phần mềm và ở tầng kiến trúc (architecture level). Dưới đây là một biểu đồ so sánh giữa monolith và microservice:

for less-complex systems, the extra baggage required to manage microservices reduces productivity

什么时候应该使用微服务？



but remember the skill of the team will outweigh any monolith/microservice choice

1.2.3 Các thuộc tính của mô hình microservice

- Autonomous (tính tự trị)

1 service sẽ là 1 đơn vị chức năng, cung cấp API để thực hiện việc trao đổi, giao tiếp với các service khác

- Isolated (tính biệt lập)

1 service sẽ là 1 đơn vị triển khai. Nó có thể được chỉnh sửa, test và deployed như 1 đơn vị mà không ảnh hưởng đến những khía cạnh khác.

- Elastic

1 service là phi trạng thái (stateless) vì vậy nó có thể scale tùy ý khi cần thiết.

- Resilient

1 microservice sẽ được thiết kế để chấp nhận các lỗi, các rủi ro có thể xảy ra, các lỗi này là các lỗi có thể chấp nhận được

- Responsive

respond cho các request trong khoảng thời gian hợp lý.

- Intelligent

Tính thông minh ở đây tức là muốn nhắc đến việc hệ thống có thể tìm thấy các endpoint của các microservice đã được đăng kí.

- Message Oriented

Mô hình micro-service hoạt động dựa trên giao thức HTTP hoặc message bus để tạo nên sự giao tiếp giữa các service. Điều này đảm bảo tính loose coupling, tính biệt lập và có thể cung cấp lỗi dưới dạng message

- Programmable

Cung cấp API's cho phép truy cập bởi developer và administrator.

- Composable

Bao gồm nhiều microservices.

- Automated

Lifecycle của Microservice được quản lý thông qua automation bao gồm development, build, test, staging, production và distribution.)

2.4 Microservice advantages

- Mỗi microservice sẽ được chia nhỏ để tập trung vào một business function cụ thể hoặc business requirement.
- Microservices có thể phát triển độc lập bởi một team nhỏ có thể chỉ từ 2 đến 5 developers.
- Microservice đem lại tính loose-coupling và context riêng cho mỗi service, sẽ dễ dàng trong quá trình development cũng như deploy một cách độc lập..
- Microservices có thể phát triển với nhiều ngôn ngữ khác nhau.
- Quá trình phát triển một service sẽ trở nên dễ dàng và linh động thông qua việc sử dụng CI/CD như Travis, Jenkins, Circle CI
- 1 new member có thể dễ dàng và nhanh chóng đóng góp cho dự án
- 1 service trong mô hình micro service là tương đối nhỏ, dễ hiểu và được quản lý bởi các thành viên của 1 team nhỏ. Do đó, họ sẽ dễ dàng tập trung vào công việc, nâng cao được hiệu năng.

- Microservices cho phép tận dụng việc áp dụng những công nghệ mới vào dự án.
- Microservices chỉ gồm business logic code và không bao gồm HTML, CSS.
- Việc deploy sẽ mất ít effort cho việc configuraton.
- Dễ dàng tích hợp 3rd-party.
- Mỗi service có dung lượng lưu trữ riêng và có thể có cơ sở dữ liệu riêng.

- **2.5 Disadvantages of microservice architecture**

- Microservice architecture có thể dẫn tới việc sử dụng quá nhiều operations.
- Cần thêm kiến thức về DevOps (<http://en.wikipedia.org/wiki/DevOps>).
- Effort của team có thể sẽ phải x2.
- Distributed systems (hệ thống phân tán) phức tạp và khó quản lý.
- Số lượng service càng lớn thì vấn đề về management complexity cũng tăng theo.

1.3 Networking là gì ?



NB-IoT



M2M

PULSE COUNTER	M-BUS	MODBUS	4..20 mA	ANALOG DIGITAL I/O
WATER	TEMPERATURE HUMIDITY	DOOR WINDOW	PRESENCE MOVEMENT	SMOKE
GPS TRACKERS	1-WIRE	RS232	RS485	CUSTOM GATEWAYS

1.3.1 Wired & Short Range wireless networks Mạng dây và Không dây tầm ngắn

Tùy chọn đầu tiên là nối dây các cảm biến. Hoặc sử dụng mạng không dây tầm ngắn. Nhưng tất nhiên, nó không cung cấp phạm vi toàn cầu và do đó rất hạn chế. Trong bối cảnh của mạng không dây, chúng ta có thể đề cập đến:

- WIFI: tiêu thụ năng lượng lớn, nó không cho phép hoạt động thực tế trên pin và không thích nghi tốt với môi trường công nghiệp.
- Bluetooth, ZigBee, Z-Wave

1.3.2 M2M – 2G, 3G, 4G & 5G networks

Until now, when it comes to connecting an object (a machine, a vehicle ...) to know its status, its location, or interact with it, the only global solution – a global geographical coverage, national, even international – consisted of equipping it with a SIM card, and using the GPRS / 3G network of a telecom operator. The problems of this technology, commonly called M2M (Machine to Machine), are known:

Cho đến nay, khi nói đến việc kết nối một đối tượng (máy móc, phương tiện...) để biết trạng thái, vị trí của nó hoặc tương tác với nó, giải pháp toàn cầu duy nhất -

phạm vi địa lý toàn cầu, quốc gia, thậm chí quốc tế - bao gồm trang bị bằng thẻ SIM và sử dụng mạng GPRS / 3G của nhà khai thác viễn thông. Các vấn đề của công nghệ này, thường được gọi là M2M (Machine to Machine), được biết đến:

- Expensive equipment: difficult to find a 3G modem less than 20-30 Euro ...
- Very high energy consumption, requiring either a permanent power source (and therefore a wiring) or a powerful battery, making the size of the sensor incompatible with many use cases
- Large footprint
- A subscription cost, for connectivity, not insignificant, related to the need for a radio network (antennas operators) dense and expensive.

It is therefore difficult, with the M2M, to connect small objects, or not having an electric source. Or to accept that the sensor costs more than the connected object itself ... Not to mention that it is undesirable for health, given the power of emissions, to carry a connected object in 3G, 24 hours a day.

Thiết bị đắt tiền: khó tìm được modem 3G dưới 20-30 Euro...

- Tiêu thụ năng lượng rất cao, yêu cầu nguồn điện cố định (và do đó là hệ thống dây điện) hoặc pin mạnh, khiến kích thước của cảm biến không tương thích với nhiều trường hợp sử dụng
- Dấu chân lớn
- Chi phí thuê bao, cho kết nối, không đáng kể, liên quan đến nhu cầu về mạng vô tuyến (các nhà khai thác ăng-ten) dày đặc và đắt tiền.

Tiền thiết bị: khó tìm được modem 3G dưới 20-30 Euro...

- Tiêu thụ năng lượng rất cao, cố định nguồn yêu cầu (và do đó là dây điện hệ thống) hoặc ghim mạnh, làm cho kích thước của cảm biến không tương thích với nhiều trường hợp sử dụng

- Dấu lớn chân
- Chi phí thuê bao, cho kết nối, không đáng kể, liên quan đến nhu cầu về mạng vô tuyến (các nhà khai thác ăng-ten) dày đặc và thanh toán tiền.

1.3.3 LPWAN – Low Power Wide Area Networks

It is in this context that LPWAN – Low Power Wide Area Network – technologies have emerged. The objectives of this technology are to propose sensors

- Cheap – in the end only a few euros ...
- Small sizes – a few centimeters away, or less ...
- Can operate on battery, with great autonomy (5-10years!)
- Optimized to communicate for very small data rates, for a few Euro cents a month

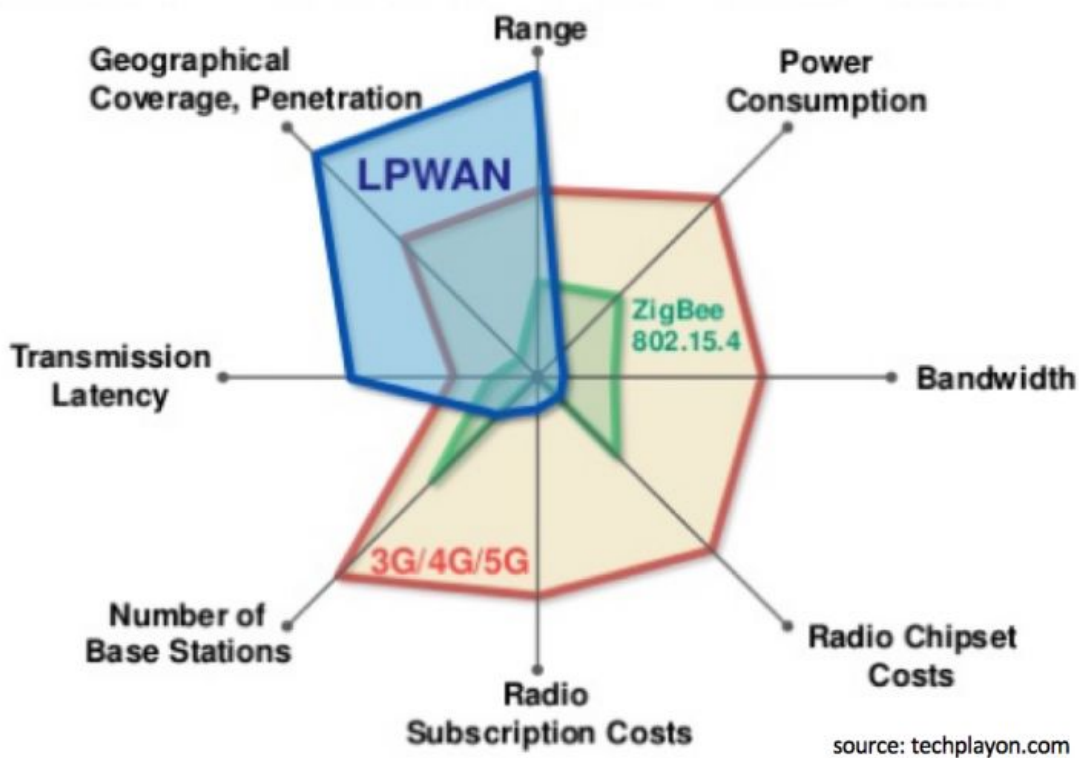
If the LPWAN technology is not recent, with already implemented in the 90's (for example the Alarm Net network in the United States), it will be necessary to wait for the creation of SIGFOX (2009) and the launch of its first network in 2012, in France, so that it begins to be structured and to become popularized.

Chính trong bối cảnh đó, công nghệ LPWAN - Mạng diện rộng công suất thấp - đã xuất hiện. Mục tiêu của công nghệ này là đề xuất các cảm biến

- Rẻ - cuối cùng chỉ vài euro...
- Kích thước nhỏ - cách vài cm, hoặc ít hơn...
- Có thể hoạt động bằng pin, với quyền tự chủ tuyệt vời (5-10 năm!)
- Được tối ưu hóa để liên lạc với tốc độ dữ liệu rất nhỏ, với vài xu Euro một tháng

Nếu công nghệ LPWAN không phải là gần đây, đã được triển khai vào những năm 90 (ví dụ mạng Alarm Net ở Hoa Kỳ), thì cần phải đợi sự ra đời của SIGFOX (2009)

và ra mắt mạng đầu tiên vào năm 2012 , ở Pháp, để nó bắt đầu được cấu trúc và trở nên phổ biến.



The Main Players in the LPWAN – Low Power Wide Area Network – for IOT projects are: **SigFox** and **LoRaWAN**. Other technologies, such as **RPMA (Ingenu)**, **Weightless**, **Strij (Russia)**, aren't so popular yet. NB-IOT is also appearing...

Các phần tử chính trong LPWAN - Mạng diện rộng công suất thấp - cho các dự án IOT là: SigFox và LoRaWAN. Các công nghệ khác, chẳng hạn như RPMA (Ingenu), Weightless, Strij (Nga), chưa quá phổ biến. NB-IOT cũng đang xuất hiện...

1.3.4 SigFox & LoRaWAN

The two most used technology players today are SigFox and LoRa. If there are different techniques still both solutions (SigFox is the Ultra Narrow Band), from a

technical point of view, the two technologies are relatively equivalent. SigFox and LoRaWAN operate on a **free ISM (Industrial, Scientific and Medical) frequency band**, so there is no need for a license. If this frequency band is free, it nevertheless requires compliance with a rule, which is not to use more than 1% of the bandwidth, which explains the limitation of the number of data that can be transferred ...

The principle of operation is that the sensor (device) is in sleep mode most of the time (to save the battery), and wakes up at regular intervals – usually, at most every 10 minutes – to transmit the measured data . 12-byte data packets that are broadcast and retrieved by all visible Gateways. This data is then transmitted over the Internet to an IOT platform that will process the data.

Hai trình phát công nghệ được sử dụng nhiều nhất hiện nay là SigFox và LoRa. Nếu có các kỹ thuật khác nhau vẫn là cả hai giải pháp (SigFox là Băng tần siêu hẹp), từ quan điểm kỹ thuật, hai công nghệ này tương đối tương đương nhau. **, vì vậy không cần phải có giấy phép. Nếu băng tần này là miễn phí, tuy nhiên, nó yêu cầu tuân thủ một quy tắc, đó là không sử dụng nhiều hơn 1% băng thông, điều này giải thích giới hạn về số lượng dữ liệu có thể được truyền...

Nguyên tắc hoạt động là phần lớn thời gian cảm biến (thiết bị) ở chế độ ngủ (để tiết kiệm pin) và thức dậy theo chu kỳ - thường là nhiều nhất 10 phút một lần - để truyền dữ liệu đo được. Các gói dữ liệu 12 byte được phát và truy xuất bởi tất cả các Cổng có thể nhìn thấy. Dữ liệu này sau đó được truyền qua Internet tới một nền tảng IOT sẽ xử lý dữ liệu.



One global Operator. Available today in more than 30 countries. Partnership at the country level with one company for the local roll-out. No constraint on the SigFox radio chipset.

Uplink (data from the device to the network: 140 messages of 12 bytes / day

Downlink (data from the network to the device): 4 messages / day

Key Strengths: a global network (with roaming “by design”)

Một nhà điều hành toàn cầu. Hiện có mặt tại hơn 30 quốc gia. Quan hệ đối tác ở cấp quốc gia với một công ty để triển khai tại địa phương. Không có ràng buộc đối với chipset vô tuyến SigFox.

Đường lên (dữ liệu từ thiết bị vào mạng: 140 tin nhắn 12 byte / ngày

Đường xuống (dữ liệu từ mạng về thiết bị): 4 tin nhắn / ngày

Điểm mạnh chính: mạng toàn cầu (với chuyển vùng “theo thiết kế”)



An Open Standard (LoRa Alliance) for an IOT Network. Everybody can deploy and operate a public or private LoRaWAN Network. But it is mandatory to use the Semtech chipset (radio modem)

Uplink (data from the device to the network: 140 messages of 12 bytes / day

Downlink: more flexible

Key Strengths: Possibility to build its own LoRaWAN private networks. Tens of Thousands of networks in the world

Tiêu chuẩn mở (Liên minh LoRa) cho Mạng IOT. Mọi người đều có thể triển khai và vận hành Mạng LoRaWAN công cộng hoặc riêng tư. Nhưng bắt buộc phải sử dụng chipset Semtech (modem radio)

Đường lên (dữ liệu từ thiết bị vào mạng: 140 tin nhắn 12 byte / ngày

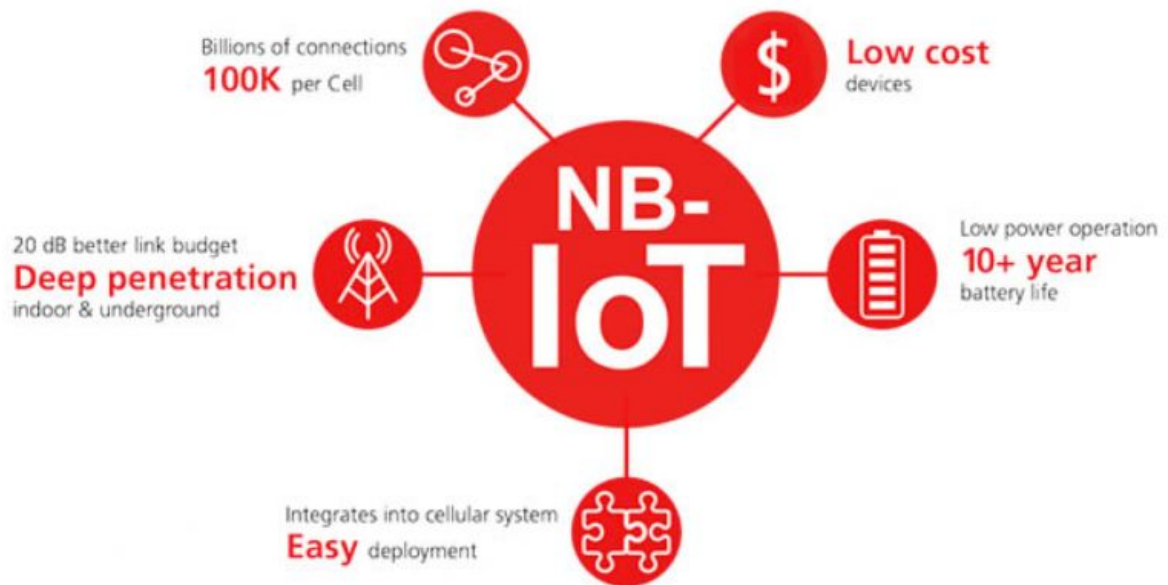
Đường xuống: linh hoạt hơn

Điểm mạnh chính: Khả năng xây dựng mạng riêng LoRaWAN của riêng mình. Hàng chục nghìn mạng trên thế giới

1.3.5 NB-IOT (Narrow Band IOT)

NB-IOT, for “Narrow Band IOT”, is the answer of traditional GSM operators to also offer a LPWAN network. This is a recent specification, an evolution of 4G (LTE) that offers a low power consumption solution, with limited throughput, based on an update of the current 4G network.

NB-IOT, cho “IOT băng tần hẹp”, là câu trả lời của các nhà khai thác GSM truyền thống cũng cung cấp mạng LPWAN. Đây là thông số kỹ thuật gần đây, một sự phát triển của 4G (LTE) cung cấp giải pháp tiêu thụ điện năng thấp, với thông lượng hạn chế, dựa trên bản cập nhật của mạng 4G hiện tại.



If this solution offers the advantage of global country coverage (since it is only an update of the current network), it is likely that the energy consumption will not be as low, and that the cost of the sensors will be higher. Moreover, it is impossible to build a private network based on NB-IOT. It is in any case an interesting alternative, but far from perfect.

Nếu giải pháp này mang lại lợi thế về phạm vi phủ sóng quốc gia toàn cầu (vì nó chỉ là bản cập nhật của mạng hiện tại), thì có khả năng mức tiêu thụ năng lượng sẽ không thấp và chi phí của các cảm biến sẽ cao hơn. Hơn nữa, không thể xây dựng một mạng riêng dựa trên NB-IOT. Trong mọi trường hợp, nó là một sự thay thế thú vị, nhưng không hoàn hảo.

1.4 Tại sao tôi lại đề cập đến vấn đề này trong khi nhiều người đã làm trước đó, và có gì khác biệt với các công nghệ trước đó ? Tôi định làm cái gì ?

Chương 2: THINGSBOARD - IoT Platform

2.1 Thingsboard là gì ?

2.1.1 What is ThingsBoard?

ThingsBoard là một nền tảng IoT mã nguồn mở cho phép phát triển, quản lý và mở rộng nhanh chóng các dự án IoT. Mục tiêu của chúng tôi là cung cấp giải pháp đám mây IoT sẵn có hoặc giải pháp tại chỗ sẽ kích hoạt cơ sở hạ tầng phía máy chủ cho các ứng dụng IoT của bạn.

2.1.2 Đặc trưng

Với ThingsBoard, bạn có thể:

- Cung cấp thiết bị, tài sản và khách hàng và xác định mối quan hệ giữa chúng.
- Thu thập và trực quan hóa dữ liệu từ các thiết bị và tài sản.
- Phân tích phép đo từ xa đến và kích hoạt cảnh báo với xử lý sự kiện phức tạp.
- Điều khiển thiết bị của bạn bằng cách gọi thủ tục từ xa (RPC).
- Xây dựng luồng công việc dựa trên sự kiện vòng đời thiết bị, sự kiện REST API, yêu cầu RPC, v.v.
- Thiết kế trang tổng quan năng động và đáp ứng, đồng thời giới thiệu thiết bị hoặc nội dung và thông tin chi tiết cho khách hàng của bạn
- Bật các tính năng cụ thể cho từng trường hợp sử dụng bằng cách sử dụng chuỗi quy tắc có thể tùy chỉnh.
- Đẩy dữ liệu thiết bị sang hệ thống khác.
- Nhiều hơn nữa...

ThingsBoard được thiết kế để:

- có thể mở rộng **: nền tảng có thể mở rộng theo chiều ngang, được xây dựng bằng các công nghệ mã nguồn mở hàng đầu.

- chịu lỗi **: không có điểm lỗi duy nhất, mọi nút trong cụm đều giống hệt nhau.
- mạnh mẽ và hiệu quả **: một nút máy chủ duy nhất có thể xử lý hàng chục hoặc thậm chí hàng trăm nghìn thiết bị tùy thuộc vào trường hợp sử dụng. Cụm ThingsBoard có thể xử lý hàng triệu thiết bị.
- tùy chỉnh **: dễ dàng thêm chức năng mới với các widget có thể tùy chỉnh và các nút công cụ quy tắc.
- bền **: không bao giờ mất dữ liệu của bạn.

2.2 Các công nghệ thingsboard sử dụng ?

2.1.1 Tính năng của Phiên bản Cộng đồng

- [Thuộc tính] (<https://thingsboard.io/docs/user-guide/attributes/>) ** - khả năng của nền tảng để gán các thuộc tính khóa-giá trị tùy chỉnh cho các thực thể của bạn (ví dụ: cấu hình, xử lý dữ liệu, thông số hiển thị).
- [Đo từ xa] (<https://thingsboard.io/docs/user-guide/telemetry/>) ** - API để thu thập dữ liệu chuỗi thời gian và các trường hợp sử dụng liên quan.
- [Thực thể và quan hệ] (<https://thingsboard.io/docs/user-guide/rpc/>) ** - khả năng của nền tảng để lập mô hình các đối tượng trong thế giới vật lý (ví dụ: thiết bị và nội dung) và mối quan hệ giữa chúng.
- [Trực quan hóa dữ liệu] (<https://thingsboard.io/docs/guides#AnchorIDDataVisualization>) ** - bao gồm các khả năng trực quan hóa dữ liệu: Tiện ích, Trang tổng quan, Trạng thái Trang tổng quan.
- [Công cụ quy tắc] (<https://thingsboard.io/docs/user-guide/rule-engine-2-0/re-getting-started/>) ** - bao gồm xử lý dữ liệu và các hành động trên sự kiện và phép đo từ xa đến .

- [RPC] (<https://thingsboard.io/docs/user-guide/rpc/>) ** - API và tiện ích con để đẩy các lệnh từ ứng dụng và trang tổng quan của bạn sang thiết bị và ngược lại.
- [Nhật ký kiểm tra] (<https://thingsboard.io/docs/user-guide/audit-log/>) ** - theo dõi hoạt động của người dùng và sử dụng lệnh gọi API.
- [Giới hạn API] (<https://thingsboard.io/docs/user-guide/api-limits/>) ** - kiểm soát việc sử dụng API, bằng cách giới hạn số lượng yêu cầu từ một máy chủ trong một đơn vị thời gian.

2.1.2 Tính năng của Phiên bản Chuyên nghiệp

- [Dán nhãn trắng] (<https://thingsboard.io/docs/user-guide/white-labeling/>) ** - định cấu hình biểu trưng công ty hoặc sản phẩm, bảng màu và thời gian gửi thư trong 2 phút.
- [Tích hợp nền tảng] (<https://thingsboard.io/docs/user-guide/integrations/>) ** - kết nối các thiết bị sử dụng các giải pháp kết nối như NB IoT, LoRaWAN và SigFox, các định dạng trọng tải cụ thể hoặc các Nền tảng IoT khác nhau
 - [HTTP] (<https://thingsboard.io/docs/user-guide/integrations/http/>) **
 - [MQTT] (<https://thingsboard.io/docs/user-guide/integrations/mqtt/>) **
 - [OPC-UA] (<https://thingsboard.io/docs/user-guide/integrations/opc-ua/>) **

- [SigFox]
(<https://thingsboard.io/docs/user-guide/integrations/sigfox/>) **
- [ThingPark]
(<https://thingsboard.io/docs/user-guide/integrations/thingpark/>) **
- [TheThingsNetwork]
(<https://thingsboard.io/docs/user-guide/integrations/ttn/>) **
- [Trung tâm sự kiện Azure]
(<https://thingsboard.io/docs/user-guide/integrations/azure-event-hub/>) **
- [Azure IoT Hub]
(<https://thingsboard.io/docs/user-guide/integrations/azure-iot-hub/>) **
- [IBM Watson IoT]
(<https://thingsboard.io/docs/user-guide/integrations/ibm-watson-iot/>) **
- [AWS IoT]
(<https://thingsboard.io/docs/user-guide/integrations/aws-iot/>) **
- [AWS Kinesis]
(<https://thingsboard.io/docs/user-guide/integrations/aws-kinesis/>) **
- [Nhóm thiết bị & nội dung]
(<https://thingsboard.io/docs/user-guide/groups/>) ** - định cấu hình
nhiều nhóm thiết bị & nội dung tùy chỉnh.

- [Trình lập lịch] (<https://thingsboard.io/docs/user-guide/scheduler/>) ** - lập lịch các loại sự kiện khác nhau (tức là cập nhật cấu hình, tạo báo cáo, lệnh rpc) với các tùy chọn cấu hình linh hoạt.
- [Báo cáo] (<https://thingsboard.io/docs/user-guide/reporting/>) ** - tạo báo cáo bằng cách sử dụng trang tổng quan hiện có và phân phối chúng cho người dùng cuối qua email.
- [Xuất dữ liệu CSV / XLS] (<https://thingsboard.io/docs/user-guide/csv-xls-data-export/>) ** - xuất dữ liệu từ tiện ích con sang CSV hoặc XLS.
- [Lưu trữ tệp] (<https://thingsboard.io/docs/user-guide/file-storage/>) ** - khả năng lưu trữ nội dung nhị phân (tệp) trong DB.

2.1.3 Bảo vệ

Chứa mô tả về xác thực thiết bị có sẵn ** [tùy chọn] (<https://thingsboard.io/docs/user-guide/device-credentials/>) **.

2.1.4 Mẫu

Chứa danh sách các nền tảng phần cứng cụ thể ** [mẫu] (<https://thingsboard.io/docs/samples/>) **.

2.1.5 API

Chứa danh sách kết nối thiết bị và nền tảng phía máy chủ cụ thể ** [API] (<https://thingsboard.io/docs/samples/>) **.

2.1.6 Cổng IoT

Chứa tài liệu toàn diện về ThingsBoard ** [IoT Gateway] (<https://thingsboard.io/docs/iot-gateway/>) **.

2.1.7 Video hướng dẫn

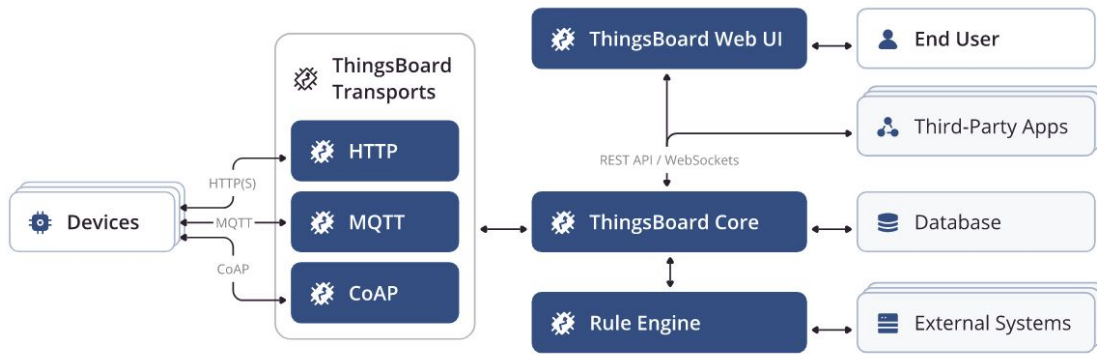
The ThingsBoard Youtube ** [kênh]
(<https://www.youtube.com/channel/UCDb9fsV-YR4JmnipAMGsVAQ/videos>) **
chứa các video hướng dẫn hữu ích bao gồm các tính năng nền tảng khác nhau.

2.1.8 Các tính năng của Trendz Analytics




- [Thời gian dự báo] (<https://thingsboard.io/docs/trendz/prediction>) ** - dự đoán hành vi của hệ thống và sử dụng dự báo
- [Các trường được tính toán] (<https://thingsboard.io/docs/trendz/calculated-fields>) ** - xác định, giám sát và so sánh các KPI tùy chỉnh
- [Bang] (<https://thingsboard.io/docs/trendz/states>) ** - theo dõi lượng thời gian thiết bị dành cho các trạng thái khác nhau
- [Lọc] (<https://thingsboard.io/docs/trendz/data-filtering>) ** - xóa và lọc dữ liệu theo bất kỳ trường nào
- [Hình ảnh trực quan nâng cao] (<https://thingsboard.io/docs/trendz/visualizations-overview>) ** - tạo bản đồ nhiệt, phân tán biểu đồ và so sánh các tiện ích
- [Nhóm và tổng hợp] (<https://thingsboard.io/docs/trendz/data-grouping-aggregation>) ** - tổng hợp dữ liệu ở các cấp độ khác nhau trong vài phút

2.3. Thingsboard với microservices

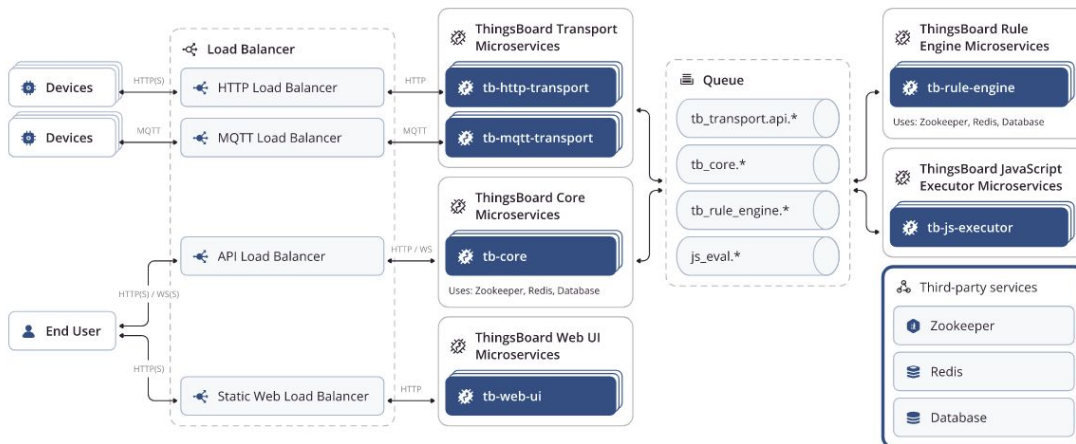
2.3.1. ThingsBoard Monolithic architecture






Legend:

-  - ThingsBoard Components: horizontally scalable and fault tolerant
-  - External Systems: Kafka, RabbitMQ, AWS SQS / SNS, Email, SMS, and other systems
-  - Databases: PostgreSQL and Timescale/Cassandra (NoSQL)

2.3.2 ThingsBoard Microservices architecture



Legend:

-  - ThingsBoard Components: horizontally scalable and fault tolerant
-  - Message Queue: Kafka, RabbitMQ, AWS SQS, Azure Service Bus, Google Pub/Sub
-  - Databases: PostgreSQL and Timescale/Cassandra (NoSQL)

2.3.2.1 Dịch vụ vi mô vận tải

ThingsBoard cung cấp các API dựa trên MQTT, HTTP và CoAP có sẵn cho các ứng dụng / chương trình cơ sở thiết bị của bạn. Mỗi API giao thức được cung cấp bởi một thành phần máy chủ riêng biệt và là một phần của “Lớp truyền tải” ThingsBoard. Danh sách đầy đủ các thành phần và các trang tài liệu tương ứng được liệt kê dưới đây:

- Dịch vụ microservice HTTP cung cấp các API thiết bị được mô tả [tại đây] (<https://thingsboard.io/docs/reference/http-api/>);
- MQTT Transport microservice cung cấp các API thiết bị được mô tả [tại đây] (<https://thingsboard.io/docs/reference/mqtt-api/>) và cũng cho phép các API cổng được mô tả [tại đây] (<https://thingsboard.io/docs/reference/gateway-mqtt-api/>);
- Dịch vụ vi mô CoAP Transport cung cấp các API thiết bị được mô tả [tại đây] (<https://thingsboard.io/docs/reference/coap-api/>).

Mỗi máy chủ truyền tải được liệt kê ở trên giao tiếp với các dịch vụ chính của ThingsBoard Node bằng Kafka. [Apache Kafka] (<https://kafka.apache.org/>) là một nền tảng phát trực tuyến và hàng đợi tin nhắn liên tục phân tán, đáng tin cậy và có thể mở rộng.

Các thư được gửi đến Kafka được tuần tự hóa bằng [bộ đệm giao thức] (<https://developers.google.com/protocol-buffers/>) với định nghĩa thư có sẵn [tại đây] (<https://github.com/thingsboard/thingsboard/blob/master/common/transport/transport-api/src/main/proto/transport.proto>).

- Lưu ý **: Bắt đầu v2.5, ThingsBoard PE sẽ hỗ trợ triển khai hàng đợi thay thế: Amazon DynamoDB. Xem [lộ trình] (<https://thingsboard.io/docs/reference/roadmap>) để biết thêm chi tiết.

Có hai chủ đề chính được sử dụng bởi microservices của lớp vận chuyển.

Chủ đề đầu tiên “tb.transport.api.requests” được sử dụng để thực thi các yêu cầu API tồn tại trong thời gian ngắn nhằm kiểm tra thông tin đăng nhập thiết bị hoặc tạo thiết bị thay mặt cho công. Phản hồi cho các yêu cầu này được gửi đến chủ đề cụ thể cho từng dịch vụ vì mô vận tải. Tiền tố của chủ đề “gọi lại” như vậy là “tb.transport.api.responses” theo mặc định.

Chủ đề thứ hai “tb.rule-engine” được sử dụng để lưu trữ tất cả các tin nhắn đo từ xa đến từ các thiết bị cho đến khi chúng không được xử lý bởi rule engine. Trong trường hợp (các) nút của công cụ quy tắc bị lỗi, các thông báo sẽ vẫn tồn tại và có sẵn để xử lý sau.

Bạn có thể xem một phần của tệp cấu hình để chỉ định các thuộc tính đó bên dưới:

```
transport: type: "${TRANSPORT_TYPE:local}" *# local or remote*remote:  
transport_api: requests_topic:  
"${TB_TRANSPORT_API_REQUEST_TOPIC:tb.transport.api.requests}"  
responses_topic:  
"${TB_TRANSPORT_API_RESPONSE_TOPIC:tb.transport.api.responses}"  
rule_engine: topic: "${TB_RULE_ENGINE_TOPIC:tb.rule-engine}"
```

Vì ThingsBoard sử dụng giao thức giao tiếp rất đơn giản giữa các dịch vụ vận tải và cốt lõi, nên việc triển khai hỗ trợ giao thức truyền tải tùy chỉnh là khá dễ dàng, ví dụ: CSV qua TCP đơn giản, tải trọng nhị phân qua UDP, v.v. Chúng tôi khuyên bạn nên xem lại việc triển khai truyền tải hiện có để bắt đầu hoặc liên hệ với chúng tôi nếu bạn cần bất kỳ trợ giúp nào.

2.3.2.2 Web UI Microservices

ThingsBoard cung cấp một thành phần nhẹ được viết bằng khung Express.js để lưu trữ nội dung ui web tĩnh. Những thành phần đó hoàn toàn không có trạng thái và không có nhiều cấu hình.

2.3.2.3 JavaScript Executor Microservices

Công cụ quy tắc ThingsBoard cho phép người dùng chỉ định các chức năng javascript tùy chỉnh để phân tích cú pháp, lọc và chuyển đổi thông báo. Vì các hàm đó là do người dùng định nghĩa, chúng ta cần thực thi chúng trong một ngữ cảnh riêng biệt để tránh ảnh hưởng đến quá trình xử lý chính. ThingsBoard cung cấp một thành phần nhẹ được viết bằng Node.js để thực thi các chức năng JavaScript do người dùng xác định từ xa để tách chúng khỏi các thành phần công cụ quy tắc cốt lõi.

- Lưu ý **: Ứng dụng nguyên khối ThingsBoard thực thi các chức năng do người dùng xác định trong công cụ JS nhưng java, không cho phép cô lập việc tiêu thụ tài nguyên.

Chúng tôi khuyên bạn nên khởi chạy hơn 20 Trình thực thi JavaScript riêng biệt sẽ cho phép mức đồng thời nhất định và cân bằng tải các yêu cầu thực thi JS. Mỗi microservice sẽ đăng ký chủ đề kafka “js.eval.requests” như một phần của nhóm người tiêu dùng đơn lẻ để bật cân bằng tải. Các yêu cầu cho cùng một tập lệnh được chuyển tiếp đến cùng một trình thực thi JS bằng cách sử dụng phân vùng Kafka tích hợp sẵn theo khóa (khóa là id nút tập lệnh / quy tắc).

Có thể xác định số lượng yêu cầu thực thi JS tối đa đang chờ xử lý và thời gian chờ yêu cầu tối đa để tránh việc thực thi JS đơn lẻ chặn microservice của JS. Mỗi dịch

vụ cốt lõi ThingsBoard có danh sách đen riêng cho các hàm JS và sẽ không gọi hàm bị chặn quá 3 lần (theo mặc định).

2.3.2.4 ThingsBoard Node

Nút ThingsBoard là một dịch vụ cốt lõi được viết bằng Java chịu trách nhiệm xử lý:

- Lệnh gọi [REST API] (<https://thingsboard.io/docs/reference/rest-api/>);
- WebSocket [đăng ký] (<https://thingsboard.io/docs/user-guide/telemetry/#websocket-api>) về phép đo từ xa đối tượng và các thay đổi thuộc tính;
- Xử lý tin nhắn qua [rule engine] (<https://thingsboard.io/docs/user-guide/rule-engine-2-0/re-getting-started/>);
- Thiết bị giám sát [trạng thái kết nối] (<https://thingsboard.io/docs/user-guide/device-connectivity-status/>) (hoạt động / không hoạt động).
 - Lưu ý : chuyển công cụ quy tắc sang một dịch vụ vi mô riêng biệt được lên lịch cho ThingsBoard v2.5. Xem [lộ trình] (<https://thingsboard.io/docs/reference/roadmap>) để biết thêm chi tiết.

Nút ThingsBoard sử dụng Hệ thống diễn viên để triển khai đối tượng thuê, thiết bị, chuỗi quy tắc và các tác nhân nút quy tắc. Các nút nền tảng có thể tham gia vào cụm, trong đó mỗi nút bằng nhau. Khám phá dịch vụ được thực hiện thông qua Zookeeper. Các nút ThingsBoard định tuyến các thông báo giữa nhau bằng cách sử dụng thuật toán băm nhất quán dựa trên id thực thể. Vì vậy, các thông báo cho cùng một thực thể được xử lý trên cùng một nút ThingsBoard. Nền tảng sử dụng [gRPC] (<https://grpc.io/>) để gửi tin nhắn giữa các nút ThingsBoard.

- Lưu ý **: Các tác giả của ThingsBoard cân nhắc việc chuyển từ gRPC sang Kafka trong các bản phát hành trong tương lai để trao đổi thông

điệp giữa các nút ThingsBoard. Ý tưởng chính là hy sinh các hình phạt về hiệu suất / độ trễ nhỏ để có lợi cho việc phân phối tin nhắn liên tục và đáng tin cậy và cân bằng tải tự động do các nhóm người tiêu dùng Kafka cung cấp.

2.3.2.5 Third-party

2.3.2.5.1 Kafka

[Apache Kafka] (<https://kafka.apache.org/>) là một nền tảng phần mềm xử lý luồng mã nguồn mở. ThingsBoard sử dụng Kafka để duy trì phép đo từ xa đến từ các chuyển vị HTTP / MQTT / CoAP cho đến khi nó được xử lý bởi công cụ quy tắc. ThingsBoard cũng sử dụng Kafka cho một số lệnh gọi API giữa các dịch vụ vi mô.

2.3.2.5.1 Redis

[Redis] (<https://redis.io/>) là một mã nguồn mở (được cấp phép BSD), kho lưu trữ cấu trúc dữ liệu trong bộ nhớ được ThingsBoard sử dụng để lưu vào bộ nhớ đệm. ThingsBoard lưu trữ nội dung, chế độ xem thực thể, thiết bị, thông tin đăng nhập thiết bị, phiên thiết bị và quan hệ thực thể.

2.3.2.5.2 Zookeeper

[Zookeeper] (<https://zookeeper.apache.org/>) là một máy chủ mã nguồn mở cho phép điều phối phân tán có độ tin cậy cao. ThingsBoard sử dụng Zookeeper để giải quyết các yêu cầu xử lý từ một thực thể (thiết bị, tài sản, đối tượng thuê) tới một máy chủ ThingsBoard nhất định và đảm bảo rằng chỉ một máy chủ xử lý dữ liệu từ thiết bị cụ thể tại một thời điểm duy nhất.

- Lưu ý : Zookeeper cũng được Kafka sử dụng, vì vậy hầu như không có lý do gì để sử dụng song song hai dịch vụ điều phối khác nhau (Consul, etcd).

2.3.2.5.3 HAProxy (hoặc Load Balancer khác)

Chúng tôi khuyên bạn nên sử dụng HAProxy để cân bằng tải. Bạn có thể tìm thấy cấu hình tham chiếu [haproxy.cfg] (<https://github.com/thingsboard/thingsboard/blob/release-2.5/docker/haproxy/config/haproxy.cfg>) tương ứng với sơ đồ kiến trúc bên dưới:

2.3.2.5.4 Cơ sở dữ liệu

Xem “[SQL vs NoSQL vs Hybrid?]” (<https://thingsboard.io/docs/reference/#sql-vs-nosql-vs-hybrid-database-approach>)” để biết thêm chi tiết.

**Chương 3: THIẾT KẾ GIẢI PHÁP GIẢ LẬP ỨNG DỤNG QUẢN LÝ
THƯ VIỆN TẠ QUANG BỬU**

3.1 Mục đích

3.1.1 Giám sát, quản lý, cảnh báo trong thư viện Tạ Quang Bửu

3.2 Thiết kế tổng thể

3.2.1 sensors nhiệt độ, báo cháy, báo khói

3.2.2 cameras

3.2.3 còi

3.2.4 điều hòa

3.2.5 màn hình giám sát

3.3 Thiết kế chi tiết

3.3.1 Giả lập hệ thống cảm biến, cảnh báo, camera, máy tính,

3.3.2 Thiết kế hệ thống thiết bị tương ứng thực tế

3.3.3 Thiết kế Widgets giao diện ứng dụng

3.4 Kết nối dữ liệu đến Thingsboard

3.4.1 MQTT message

3.4.2 API

3.5 Lưu trữ

3.5.1 MongoDB

3.6 Phân phối dữ liệu.

3.6.1 Phân phối tải cho server

Chương 4: KẾT LUẬN

- [] Nhờ sự tư vấn và hướng dẫn của thầy Trần Hoàng Hải mà em mới có thể hoàn thành đề tài luận văn này. Tuy đã cố gắng nhưng sẽ còn nhiều thiếu sót, mong các thầy cô và các bạn cho lời khuyên để đề tài có thể hoàn thiện hơn nữa. Em xin cảm ơn !